nRF9160

Objective Product Specification

v0.7

4418_1177 v0.7 / 2018-12-10

nRF9160 features

Features:

Microcontroller:

- ARM® Cortex® -M33
  - 243 EEMBC CoreMark score running from flash memory
  - Data watchpoint and trace (DWT), embedded trace macrocell (ETM), and instrumentation trace macrocell (ITM)
  - Serial wire debug (SWD)
  - Trace port
- 1 MB flash
- 256 kB low leakage RAM
- ARM® Trustzone®
- ARM® Cryptocell 310
- 4x SPI master/slave with EasyDMA
- 4x I2C compatible two-wire master/slave with EasyDMA
- 4x UART (CTS/RTS) with EasyDMA
- I2S with EasyDMA
- Digital microphone interface (PDM) with EasyDMA
- 4x pulse width modulator (PWM) unit with EasyDMA
- 12-bit, 200 ksps ADC with EasyDMA - eigth configurable channels with programmable gain
- 2x 32-bit timer with counter mode
- 2x real-time counter (RTC)
- Programmable peripheral interconnect (PPI)
- 32 general purpose I/O pins
- Single supply voltage: 3.0 – 5.5 V

  Note: 3.3 - 5.5 V for RF 3GPP compliancy
- All necessary clock sources integrated
- Package: 10 × 16 x 1.2 mm LGA

LTE modem:

- Transceiver and baseband
- 3GPP LTE release 13 Cat-M1 and Cat-NB1 compliant
- 3GPP LTE release 14 Cat-NB1 and Cat-NB2 compliant
- RF transceiver for global coverage
  - 700-2200 MHz
  - Up to 23 dBm output power
  - -108 dBm sensitivity (LTE-M)
  - Single 50 Ω antenna interface
- LTE band support (certified):
  - Cat-M1
    - USA and Canada: B4, B13
    - Europe: B3, B20
  - Cat-NB1
    - Europe: B3, B20
- ETSI TS 102 221 compatible UICC interface
- DRX, eDRX, PSM
- 3GPP release 13 coverage enhancement
- IP v4/v6 stack
- Secure socket API

Applications:

- Sensor networks
- Logistics and asset tracking
- Smart energy
- Smart building automation
- Smart agriculture
- Industrial
- Retail and monitor devices
- Medical devices
- Wearables

Contents

1 Revision history

Date	Version	Description
December 2018	0.7	Preliminary release

2 About this document

This document is organized into chapters that are based on the modules and peripherals available in the IC.

2.1 Document status

The document status reflects the level of maturity of the document.

Document name	Description
Objective Product Specification (OPS)	Applies to document versions up to 1.0. This document contains target specifications for product development.
Product Specification (PS)	Applies to document versions 1.0 and higher. This document contains final product specifications. Nordic Semiconductor ASA reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

Table 1: Defined document names

2.2 Peripheral chapters

Every peripheral has a unique capitalized name or an abbreviation of its name, e.g. TIMER, used for identification and reference. This name is used in chapter headings and references, and it will appear in the ARM® Cortex® Microcontroller Software Interface Standard (CMSIS) hardware abstraction layer to identify the peripheral.

The peripheral instance name, which is different from the peripheral name, is constructed using the peripheral name followed by a numbered postfix, starting with 0, for example, TIMER0. A postfix is normally only used if a peripheral can be instantiated more than once. The peripheral instance name is also used in the CMSIS to identify the peripheral instance.

The chapters describing peripherals may include the following information:

- A detailed functional description of the peripheral
- Register configuration for the peripheral
- Electrical specification tables, containing performance data which apply for the operating conditions described in Peripheral chapters on page 10.

2.3 Register tables

Individual registers are described using register tables. These tables are built up of two sections. The first three colored rows describe the position and size of the different fields in the register. The following rows describe the fields in more detail.

2.3.1 Fields and values

The **Id (Field Id)** row specifies the bits that belong to the different fields in the register. If a field has enumerated values, then every value will be identified with a unique value id in the **Value Id** column.

A blank space means that the field is reserved and read as undefined, and it also must be written as `0` to secure forward compatibility. If a register is divided into more than one field, a unique field name is specified for each field in the **Field** column. The **Value Id** may be omitted in the single-bit bit fields when values can be substituted with a Boolean type enumerator range, e.g. true/false, disable(d)/enable(d), on/off, and so on.

Values are usually provided as decimal or hexadecimal. Hexadecimal values have a `0x` prefix, decimal values have no prefix.

The **Value** column can be populated in the following ways:

- Individual enumerated values, for example 1, 3, 9.
- Range of values, e.g. [0..4], indicating all values from and including 0 and 4.
- Implicit values. If no values are indicated in the **Value** column, all bit combinations are supported, or alternatively the field's translation and limitations are described in the text instead.

If two or more fields are closely related, the **Value Id**, **Value**, and **Description** may be omitted for all but the first field. Subsequent fields will indicate inheritance with '..'.

A feature marked **Deprecated** should not be used for new designs.

2.3.2 Permissions

Different fields in a register might have different access permissions enforced by hardware.

The access permission for each register field is documented in the **Access** column in the following ways:

Access	Description	Hardware behavior
RO	Read-only	Field can only be read. A write will be ignored.
WO	Write-only	Field can only be written. A read will return an undefined value.
RW	Read-write	Field can be read and written multiple times.
W1	Write-once	Field can only be written once per reset. Any subsequent write will be ignored. A read will return an undefined value.
RW1	Read-write-once	Field can be read multiple times, but only written once per reset. Any subsequent write will be ignored.

Table 2: Register field permission schemes

2.4 Registers

Register	Offset	Security	Description
DUMMY	0x514		Example of a register controlling a dummy feature

Table 3: Register overview

2.4.1 DUMMY

Address offset: 0x514

Example of a register controlling a dummy feature

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID					D	D	D	D						C	C	C								B							A	A
Reset 0x00050002	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**

ID	Acce	Field	Value ID	Value	Description	
A	RW	FIELD_A			Example of a read-write field with several enumerated values	
			Disabled	0	The example feature is disabled	
			NormalMode	1	The example feature is enabled in normal mode	
			ExtendedMode	2	The example feature is enabled along with extra functionality	
B	RW	FIELD_B			Example of a deprecated read-write field	Deprecated
			Disabled	0	The override feature is disabled	
			Enabled	1	The override feature is enabled	
C	RW	FIELD_C			Example of a read-write field with a valid range of values	
			ValidRange	[2..7]	Example of allowed values for this field	
D	RW	FIELD_D			Example of a read-write field with no restriction on the values	

3 Product overview

3.1 Introduction

The nRF9160 is a low power cellular IoT (internet of things) solution, integrating an ARM® Cortex®-M33 processor with advanced security features, a range of peripherals, as well as a complete LTE modem compliant with 3GPP LTE release 13 Cat-M1 and Cat-NB1, and 3GPP LTE release 14 Cat-NB1 and Cat-NB2 standards.

The ARM® Cortex-M33 processor is exclusively for user application software, and it offers 1 MB of flash and 256 kB of RAM dedicated to this use. The M33 application processor shares the power, clock and peripheral architecture with Nordic Semiconductor nRF51 and nRF52 Series of PAN/LAN SoCs, ensuring minimal porting efforts.

The peripheral set offers a variety of analog and digital functionality enabling single-chip implementation of a wide range of cellular IoT (internet of things) applications. ARM® TrustZone® technology, Cryptocell 310 and supporting blocks for system protection and key management, are embedded to enable advanced security needed for IoT applications.

The LTE modem integrates a very flexible transceiver supporting LTE bands from 450 MHz to 2.7 GHz (through a single 50 Ω antenna pin), and a baseband processor handling LTE Cat-M1/NB1/NB2 protocol layers L1-L3 as well as IP upper layers offering secure socket API for the application. The modem is supported by pre-qualified software builds available for free from Nordic Semiconductor.

3.2 Block diagram

This block diagram illustrates the overall system. Arrows with white heads indicate signals that share physical pins with other signals.

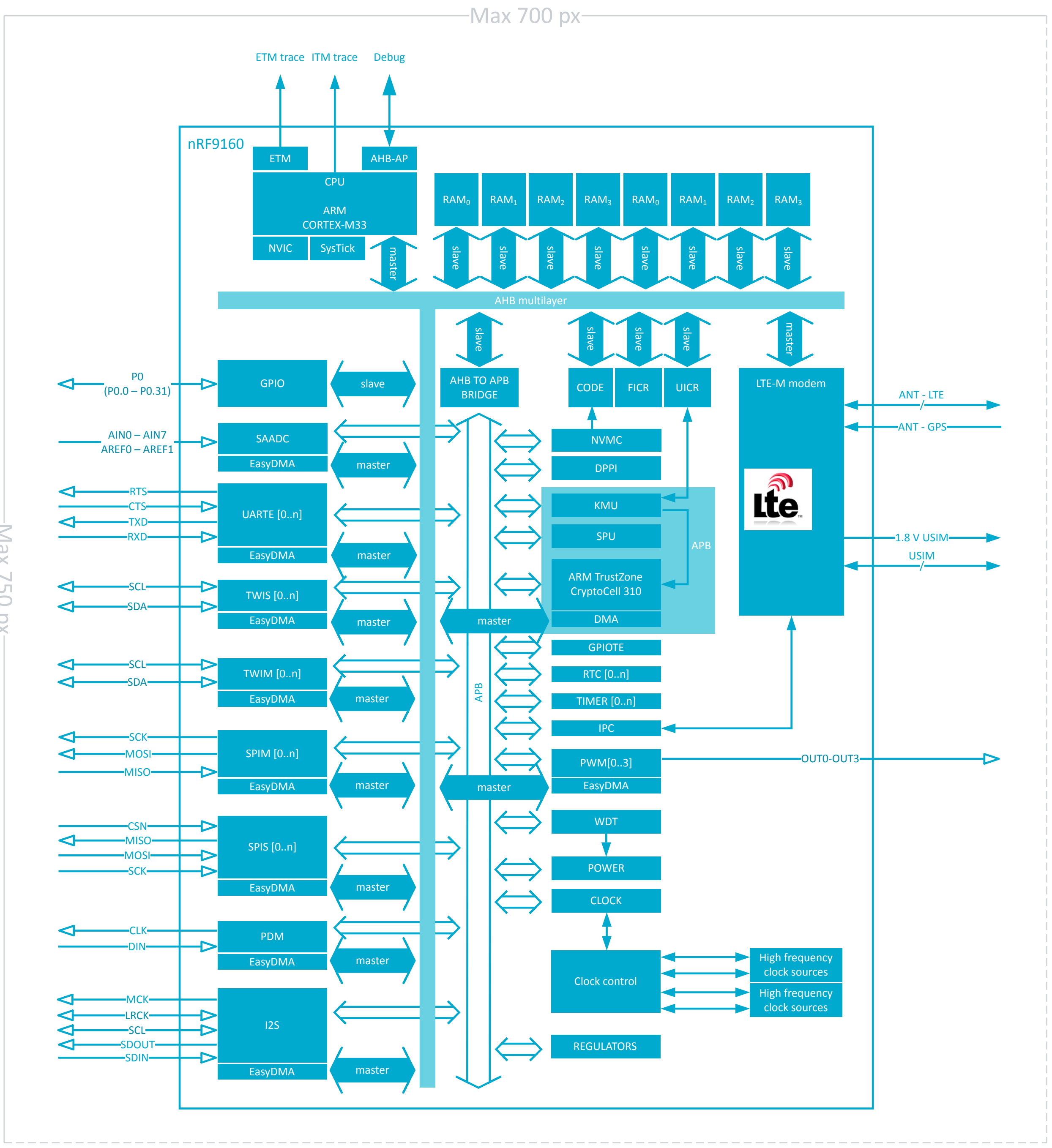


Figure 1: Block diagram

3.3 Peripheral interface

Peripherals are controlled by the CPU by writing to configuration registers and task registers. Peripheral events are indicated to the CPU by event registers and interrupts if they are configured for a given event.

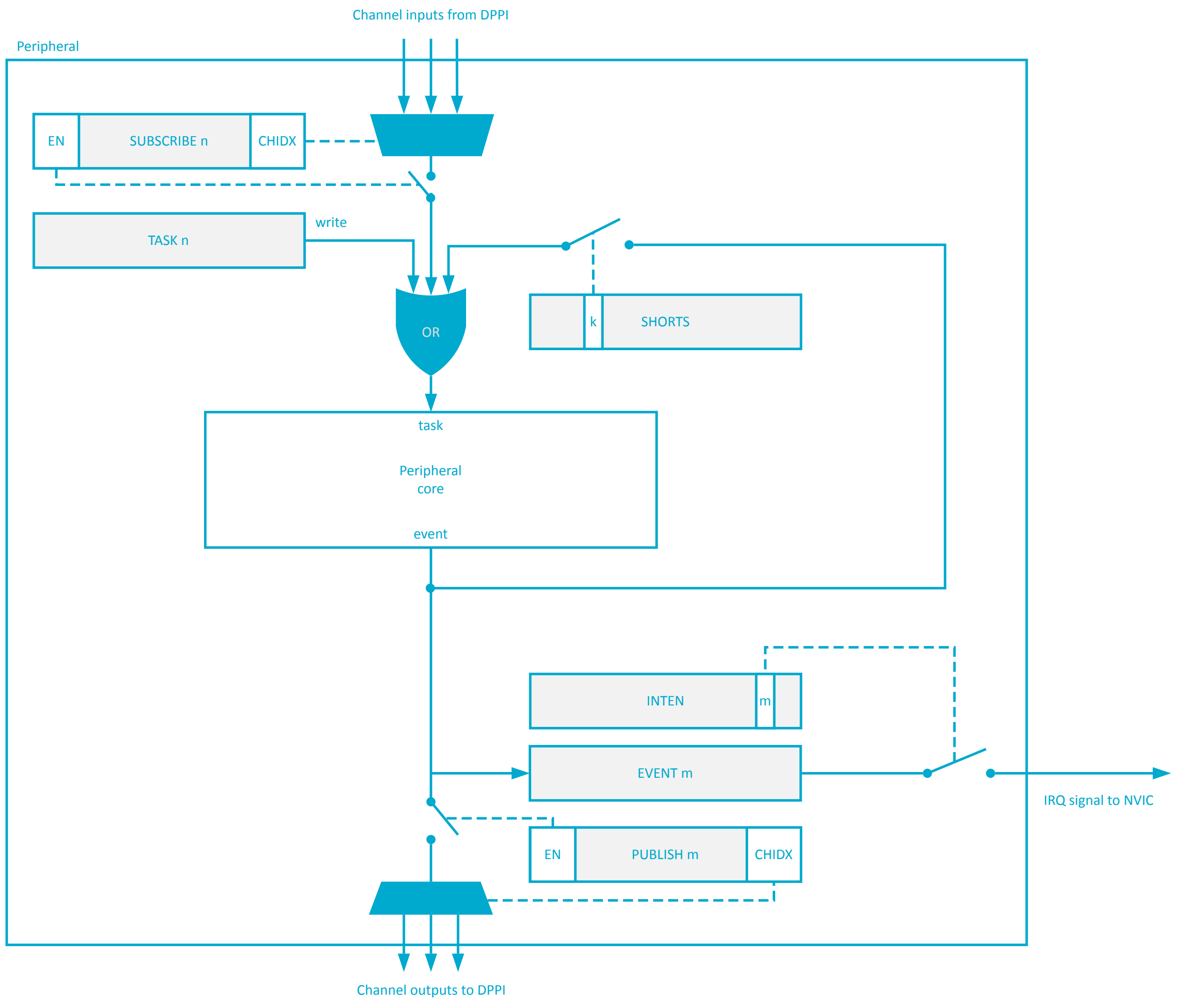


Figure 2: Tasks, events, shortcuts, publish, subscribe and interrupts

Note: For more information on DPPI channels, see DPPI - Distributed programmable peripheral interconnect on page 78.

3.3.1 Peripheral ID

Every peripheral is assigned a fixed block of 0x1000 bytes of address space, which is equal to 1024 x 32 bit registers.

See Instantiation on page 23 for more information about which peripherals are available and where they are located in the address map.

There is a direct relationship between peripheral ID and base address. For example, a peripheral with base address 0x40000000 is assigned ID=0, a peripheral with base address 0x40001000 is assigned ID=1, and a peripheral with base address 0x4001F000 is assigned ID=31.

Peripherals may share the same ID, which may impose one or more of the following limitations:

- Some peripherals share some registers or other common resources.
- Operation is mutually exclusive. Only one of the peripherals can be used at a time.
- Switching from one peripheral to another must follow a specific pattern (disable the first, then enable the second peripheral).

3.3.2 Peripherals with shared ID

In general (with the exception of ID 0), peripherals sharing an ID and base address may not be used simultaneously. The user can only enable one peripheral at the time on this specific ID.

When switching between two peripherals sharing an ID, the user should do the following to prevent unwanted behavior:

- Disable the previously used peripheral.
- Disable any publish/subscribe connection to the DPPI system for the peripheral that is being disabled.
- Clear all bits in the INTEN register, i.e. INTENCLR = 0xFFFFFFFF.
- Explicitly configure the peripheral that you are about to enable, and do not rely on configuration values that may be inherited from the peripheral that was disabled.
- Enable the now configured peripheral.

See which peripherals are sharing ID in Instantiation on page 23.

3.3.3 Peripheral registers

Most peripherals feature an ENABLE register. Unless otherwise is specified in the chapter, the peripheral registers must be configured before enabling the peripheral.

PSEL registers need to be set before a peripheral is enabled or started. Updating PSEL registers while the peripheral is running has no effect. In order to connect a peripheral to a different GPIO, the peripheral must be disabled, the PSEL register updated and the peripheral re-enabled. It takes four CPU cycles between the PSEL register update and the connection between a peripheral and a GPIO becoming effective.

Note that the peripheral must be enabled before tasks and events can be used.

Most of the register values are lost during System OFF or when a reset is triggered. Some registers will retain their values in System OFF or for some specific reset sources. These registers are marked as retained in the register description for a given peripheral. For more info on these retained registers' behavior, see chapter Reset on page 54.

3.3.4 Bit set and clear

Registers with multiple single-bit bit fields may implement the set-and-clear pattern. This pattern enables firmware to set and clear individual bits in a register without having to perform a read-modify-write operation on the main register.

This pattern is implemented using three consecutive addresses in the register map, where the main register is followed by dedicated SET and CLR registers (in that exact order).

The SET register is used to set individual bits in the main register, while the CLR register is used to clear individual bits in the main register. Writing 1 to a bit in SET or CLR register will set or clear the same bit in the main register respectively. Writing 0 to a bit in SET or CLR register has no effect. Reading the SET or CLR register returns the value of the main register.

Note: The main register may not be visible and hence not directly accessible in all cases.

3.3.5 Tasks

Tasks are used to trigger actions in a peripheral, for example to start a particular behavior. A peripheral can implement multiple tasks with each task having a separate register in that peripheral's task register group.

A task is triggered when firmware writes 1 to the task register, or when the peripheral itself or another peripheral toggles the corresponding task signal. See the figure Tasks, events, shortcuts, publish, subscribe and interrupts on page 15.

3.3.6 Events

Events are used to notify peripherals and the CPU about events that have happened, for example a state change in a peripheral. A peripheral may generate multiple events, where each event has a separate register in that peripheral's event register group.

An event is generated when the peripheral itself toggles the corresponding event signal, and the event register is updated to reflect that the event has been generated (see figure Tasks, events, shortcuts, publish, subscribe and interrupts on page 15). An event register is only cleared when firmware writes 0 to it. Events can be generated by the peripheral even when the event register is set to 1.

3.3.7 Shortcuts

A shortcut is a direct connection between an event and a task within the same peripheral. If a shortcut is enabled, the associated task is automatically triggered when its associated event is generated.

Using shortcuts is equivalent to making the connection outside the peripheral and through the DPPI. However, the propagation delay when using shortcuts is usually shorter than the propagation delay through the DPPI.

Shortcuts are predefined, which means that their connections cannot be configured by firmware. Each shortcut can be individually enabled or disabled through the shortcut register, one bit per shortcut, giving a maximum of 32 shortcuts for each peripheral.

3.3.8 Publish / Subscribe

Events and tasks from different peripherals can be connected together through the DPPI system. See Tasks, events, shortcuts, publish, subscribe and interrupts on page 15. This is done through publish / subscribe registers in each peripheral. An event can be published onto a DPPI channel by configuring the event's PUBLISH register. Similarly a task can subscribe to a DPPI channel by configuring the task's SUBSCRIBE register.

See for details.

3.3.9 Interrupts

All peripherals support interrupts. Interrupts are generated by events.

A peripheral only occupies one interrupt, and the interrupt number follows the peripheral ID. For example, the peripheral with ID=4 is connected to interrupt number 4 in the nested vectored interrupt controller (NVIC).

Using registers INTEN, INTENSET, and INTENCLR, every event generated by a peripheral can be configured to generate that peripheral's interrupt. Multiple events can be enabled to generate interrupts simultaneously. To resolve the correct interrupt source, the event registers in the event group of peripheral registers will indicate the source.

Some peripherals implement only INTENSET and INTENCLR registers, and the INTEN register is not available on those peripherals. See the individual peripheral chapters for details. In all cases, reading back the INTENSET or INTENCLR register returns the same information as in INTEN.

Each event implemented in the peripheral is associated with a specific bit position in the INTEN, INTENSET and INTENCLR registers.

The relationship between tasks, events, shortcuts, and interrupts is illustrated in figure Tasks, events, shortcuts, publish, subscribe and interrupts on page 15.

Interrupt clearing

Interrupts should always be cleared.

Clearing an interrupt by writing 0 to an event register, or disabling an interrupt using the INTENCLR register, may take a number of CPU clock cycles to take effect. This means that an interrupt may reoccur immediately, even if a new event has not come, if the program exits an interrupt handler after the interrupt is cleared or disabled but before it has taken effect.

> **Note:** To avoid an interrupt reoccurring before a new event has come, the program should perform a read from one of the peripheral registers. For example, the event register that has been cleared, or the INTENCLR register that has been used to disable the interrupt.

Care should be taken to ensure that the compiler does not remove the read operation as an optimization.

3.3.10 Secure/non-secure peripherals

For some peripherals, the security configuration can change from secure to non-secure, or vice versa. Care must be taken when changing the security configuration of a peripheral, to prevent security information leakage and ensure correct operation.

The following sequence should be followed, where applicable, when configuring and changing the security settings of a peripheral in the SPU - System protection unit on page 257:

1. Stop peripheral operation
2. Disable the peripheral
3. Remove pin connections
4. Disable DPPI connections
5. Clear sensitive registers (e.g. writing back default values)
6. Change peripheral security setting in the SPU - System protection unit on page 257
7. Re-enable the peripheral

4 Application core

4.1 CPU

The ARM® Cortex-M33 processor has a 32-bit instruction set (Thumb®-2 technology) that implements a superset of 16 and 32-bit instructions to maximize code density and performance.

This processor implements several features that enable energy-efficient arithmetic and high-performance signal processing, including:

- Digital signal processing (DSP) instructions
- Single-cycle multiply and accumulate (MAC) instructions
- Hardware divide
- 8- and 16-bit single instruction, multiple data (SIMD) instructions
- Single-precision floating-point unit (FPU)
- Memory Protection Unit (MPU)
- ARM® TrustZone® for ARMv8-M

The ARM® Cortex Microcontroller Software Interface Standard (CMSIS) hardware abstraction layer for the ARM® Cortex processor series is implemented and available for the M33 CPU.

Real-time execution is highly deterministic in thread mode, to and from sleep modes, and when handling events at configurable priority levels via the nested vectored interrupt controller (NVIC).

Executing code from internal or external flash will have a wait state penalty. The instruction cache can be enabled to minimize flash wait states when fetching instructions. For more information on cache, see Cache on page 31. The section Electrical specification on page 20 shows CPU performance parameters including the wait states in different modes, CPU current and efficiency, and processing power and efficiency based on the CoreMark® benchmark.

4.1.1 CPU and support module configuration

The ARM® Cortex®-M33 processor has a number of CPU options and support modules implemented on the device.

Option / Module	Description	Implemented
Core options		
NVIC	Nested vectored interrupt controller	
PRIORITIES	Priority bits	3
WIC	Wake-up interrupt controller	NO
Endianness	Memory system endianness	Little endian
DWT	Data watchpoint and trace	YES
Modules		
MPU_NS	Number of non-secure memory protection unit (MPU) regions	16
MPU_S	Number of secure MPU regions	16
SAU	Number of security attribution unit (SAU) regions	0, see SPU for more information about secure regions.
FPU	Floating-point unit	YES
DSP	Digital signal processing extension	YES
ARMv8-M TrustZone®	ARMv8-M security extensions	YES
CPIF	Co-processor interface	NO
ETM	Embedded trace macrocell	YES
ITM	Instrumentation trace macrocell	YES
MTB	Micro trace buffer	NO
CTI	Cross trigger interface	YES
BPU	Breakpoint unit	YES
HTM	AMBA™ AHB trace macrocell	NO

4.1.2 Electrical specification

4.1.2.1 CPU performance

The CPU clock speed is 64 MHz. Current and efficiency data is taken when in System ON and the CPU is executing the CoreMark™ benchmark. It includes power regulator and clock base currents. All other blocks are IDLE.

Symbol	Description	Min.	Typ.	Max.	Units
W_{FLASH}	CPU wait states, running from flash, cache disabled	0		4	
$W_{FLASHCACHE}$	CPU wait states, running from flash, cache enabled	0		2	
W_{RAM}	CPU wait states, running from RAM			0	
CM_{FLASH}	CoreMark[1], running from flash, cache enabled		243		Core
$CM_{FLASH/MHz}$	CoreMark per MHz, running from flash, cache enabled		3.79		CoreMark/ MHz
$CM_{FLASH/mA}$	CoreMark per mA, running from flash, cache enabled, DC/ DC		84		Core mA

4.2 Memory

The application microcontroller has embedded 1024 kB flash and 256 kB RAM for application code and data storage.

As illustrated in Memory layout on page 21, both CPU and EasyDMA are able to access RAM via the AHB multilayer interconnect. See AHB multilayer interconnect on page 47 and EasyDMA on page 44 for more information about AHB multilayer interconnect and EasyDMA respectively. The LTE modem can access all application MCU memory, but typically a small portion of RAM is dedicated to data exchange between application MCU and the modem baseband controller.

[1] Using IAR compiler

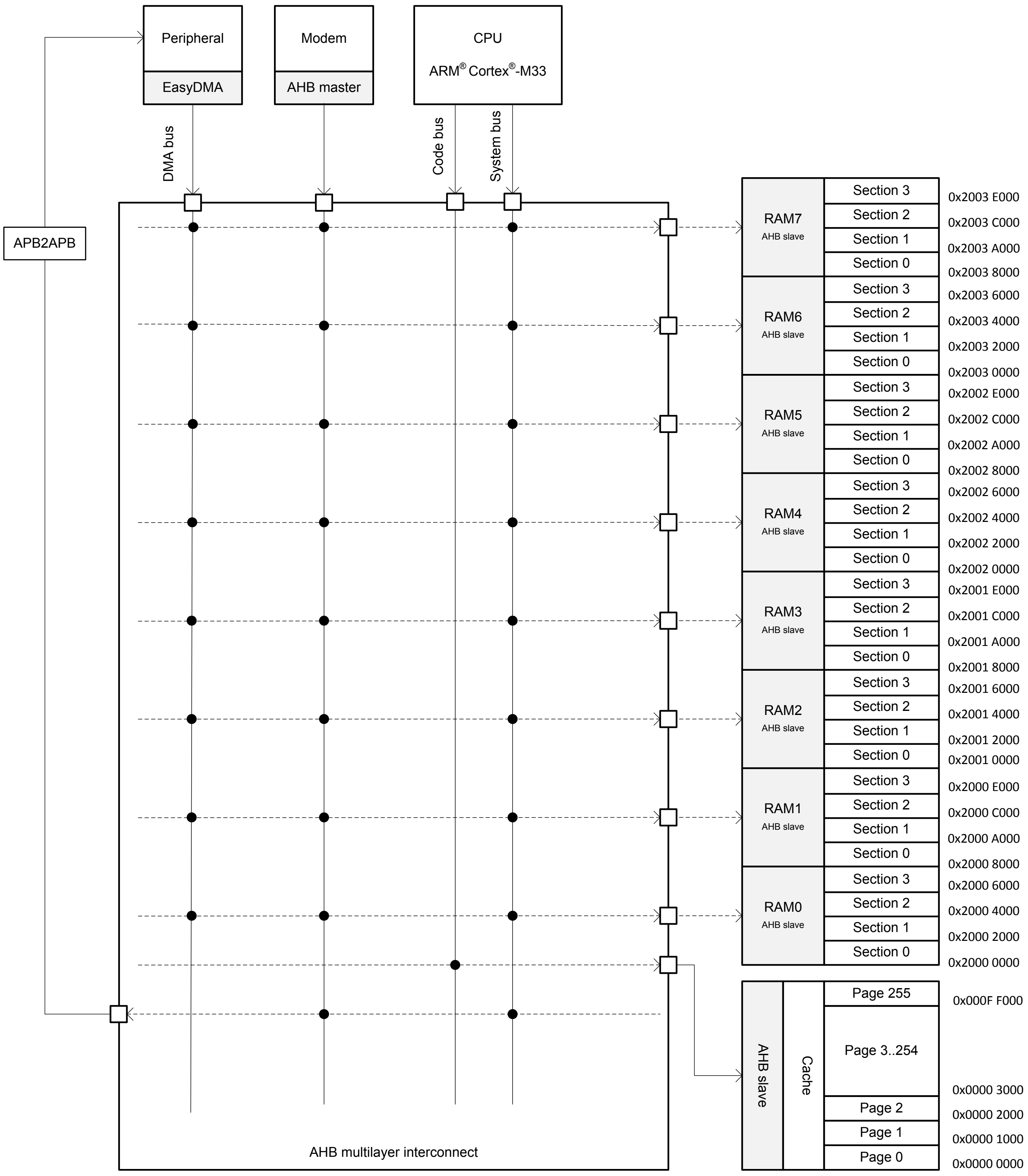


Figure 3: Memory layout

RAM - Random access memory

RAM can be read and written an unlimited number of times by the CPU and the EasyDMA.

Each RAM AHB slave is connected to one or more RAM sections. See Memory layout on page 21 for more information.

The RAM blocks power states and retention states in System ON and System OFF modes are controlled by the VMC.

Flash - Non-volatile memory

Flash can be read an unlimited number of times by the CPU and is accessible via the AHB interface connected to the CPU, see Memory layout on page 21 for more information. There are restrictions on the number of times flash can be written and erased, and also on how it can be written. Writing to flash is managed by the non-volatile memory controller (NVMC).

4.2.1 Memory map

All memory and registers are found in the same address space, as illustrated in the device memory map below.

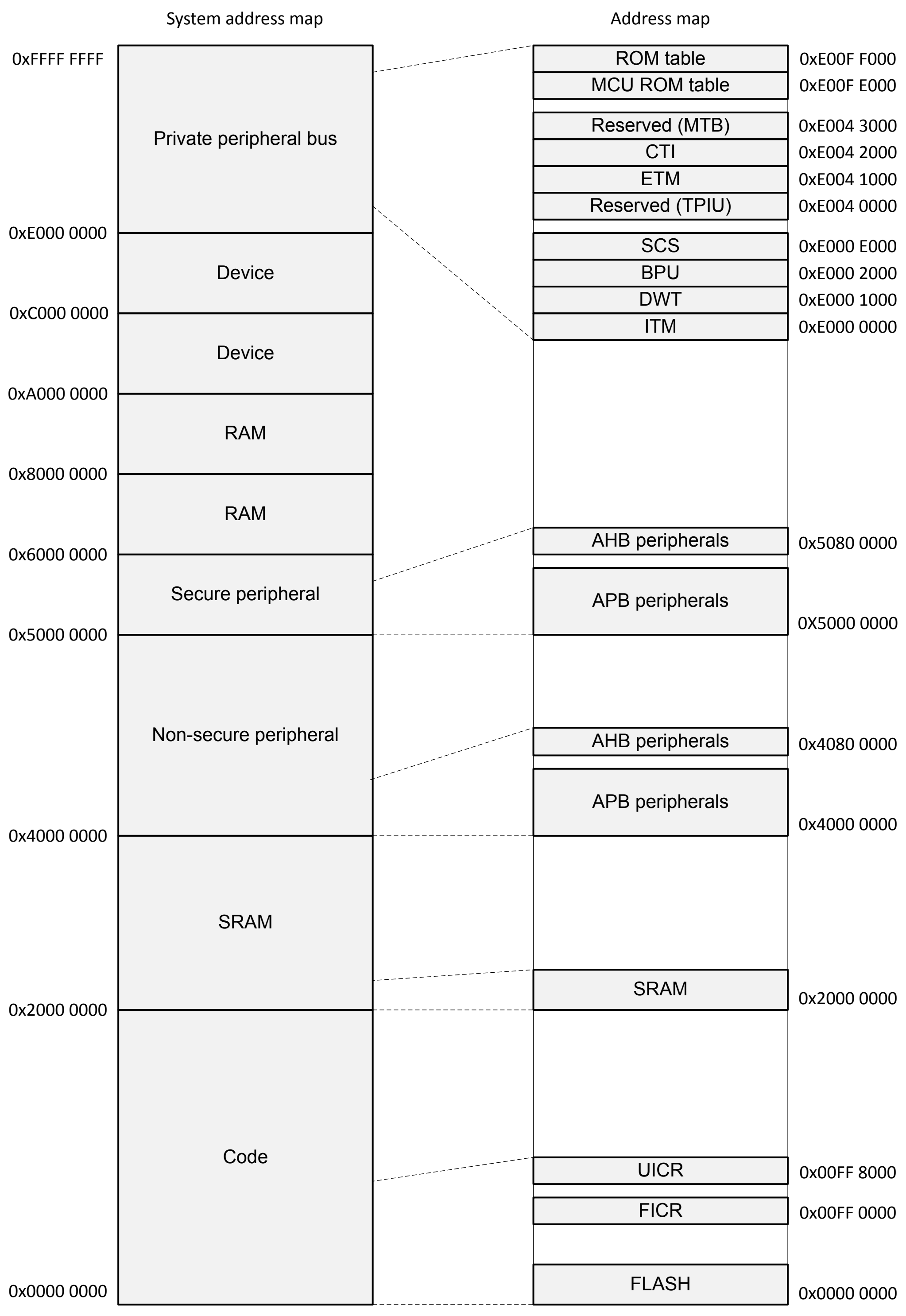


Figure 4: Memory map

Some of the registers are retained (their values kept). Read more about retained registers in Retained registers on page 54 and Reset behavior on page 55.

4.2.2 Instantiation

ID	Base address	Peripheral	Instance	Secure mapping	DMA security	Description
3	0x50003000	SPU	SPU	S	NA	System Protection Unit

ID	Base address	Peripheral	Instance	Secure mapping	DMA security	Description
4	0x50004000 0x40004000	REGULATORS	REGULATORS : S REGULATORS : NS	US	NA	Regulator configuration
5	0x50005000 0x40005000	CLOCK	CLOCK : S CLOCK : NS	US	NA	Clock control
5	0x50005000 0x40005000	POWER	POWER : S POWER : NS	US	NA	Power control
6	0x50006000	CTRLAPPERI	CTRL_AP_PERI	S	NA	CTRL-AP-PERI
8	0x50008000 0x40008000	SPIM	SPIM0 : S SPIM0 : NS	US	SA	SPI master 0
8	0x50008000 0x40008000	SPIS	SPIS0 : S SPIS0 : NS	US	SA	SPI slave 0
8	0x50008000 0x40008000	TWIM	TWIM0 : S TWIM0 : NS	US	SA	Two-wire interface master 0
8	0x50008000 0x40008000	TWIS	TWIS0 : S TWIS0 : NS	US	SA	Two-wire interface slave 0
8	0x50008000 0x40008000	UARTE	UARTE0 : S UARTE0 : NS	US	SA	Universal asynchronous receiver/transmitter with EasyDMA 0
9	0x50009000 0x40009000	SPIM	SPIM1 : S SPIM1 : NS	US	SA	SPI master 1
9	0x50009000 0x40009000	SPIS	SPIS1 : S SPIS1 : NS	US	SA	SPI slave 1
9	0x50009000 0x40009000	TWIM	TWIM1 : S TWIM1 : NS	US	SA	Two-wire interface master 1
9	0x50009000 0x40009000	TWIS	TWIS1 : S TWIS1 : NS	US	SA	Two-wire interface slave 1
9	0x50009000 0x40009000	UARTE	UARTE1 : S UARTE1 : NS	US	SA	Universal asynchronous receiver/transmitter with EasyDMA 1
10	0x5000A000 0x4000A000	SPIM	SPIM2 : S SPIM2 : NS	US	SA	SPI master 2
10	0x5000A000 0x4000A000	SPIS	SPIS2 : S SPIS2 : NS	US	SA	SPI slave 2
10	0x5000A000 0x4000A000	TWIM	TWIM2 : S TWIM2 : NS	US	SA	Two-wire interface master 2
10	0x5000A000 0x4000A000	TWIS	TWIS2 : S TWIS2 : NS	US	SA	Two-wire interface slave 2
10	0x5000A000 0x4000A000	UARTE	UARTE2 : S UARTE2 : NS	US	SA	Universal asynchronous receiver/transmitter with EasyDMA 2
11	0x5000B000 0x4000B000	SPIM	SPIM3 : S SPIM3 : NS	US	SA	SPI master 3
11	0x5000B000 0x4000B000	SPIS	SPIS3 : S SPIS3 : NS	US	SA	SPI slave 3
11	0x5000B000 0x4000B000	TWIM	TWIM3 : S TWIM3 : NS	US	SA	Two-wire interface master 3
11	0x5000B000 0x4000B000	TWIS	TWIS3 : S TWIS3 : NS	US	SA	Two-wire interface slave 3
11	0x5000B000 0x4000B000	UARTE	UARTE3 : S UARTE3 : NS	US	SA	Universal asynchronous receiver/transmitter with EasyDMA 3
13	0x5000D000	GPIOTE	GPIOTE0	S	NA	Secure GPIO tasks and events
14	0x5000E000 0x4000E000	SAADC	SAADC : S SAADC : NS	US	SA	Analog to digital converter
15	0x5000F000 0x4000F000	TIMER	TIMER0 : S TIMER0 : NS	US	NA	Timer 0

ID	Base address	Peripheral	Instance	Secure mapping	DMA security	Description
16	0x50010000 0x40010000	TIMER	TIMER1 : S TIMER1 : NS	US	NA	Timer 1
17	0x50011000 0x40011000	TIMER	TIMER2 : S TIMER2 : NS	US	NA	Timer 2
20	0x50014000 0x40014000	RTC	RTC0 : S RTC0 : NS	US	NA	Real time counter 0
21	0x50015000 0x40015000	RTC	RTC1 : S RTC1 : NS	US	NA	Real time counter 1
23	0x50017000 0x40017000	DPPIC	DPPIC : S DPPIC : NS	SPLIT	NA	DPPI controller
24	0x50018000 0x40018000	WDT	WDT : S WDT : NS	US	NA	Watchdog timer
27	0x5001B000 0x4001B000	EGU	EGU0 : S EGU0 : NS	US	NA	Event generator unit 0
28	0x5001C000 0x4001C000	EGU	EGU1 : S EGU1 : NS	US	NA	Event generator unit 1
29	0x5001D000 0x4001D000	EGU	EGU2 : S EGU2 : NS	US	NA	Event generator unit 2
30	0x5001E000 0x4001E000	EGU	EGU3 : S EGU3 : NS	US	NA	Event generator unit 3
31	0x5001F000 0x4001F000	EGU	EGU4 : S EGU4 : NS	US	NA	Event generator unit 4
32	0x50020000 0x40020000	EGU	EGU5 : S EGU5 : NS	US	NA	Event generator unit 5
33	0x50021000 0x40021000	PWM	PWM0 : S PWM0 : NS	US	SA	Pulse width modulation unit 0
34	0x50022000 0x40022000	PWM	PWM1 : S PWM1 : NS	US	SA	Pulse width modulation unit 1
35	0x50023000 0x40023000	PWM	PWM2 : S PWM2 : NS	US	SA	Pulse width modulation unit 2
36	0x50024000 0x40024000	PWM	PWM3 : S PWM3 : NS	US	SA	Pulse width modulation unit 3
38	0x50026000 0x40026000	PDM	PDM : S PDM : NS	US	SA	Pulse density modulation (digital microphone) interface
40	0x50028000 0x40028000	I2S	I2S : S I2S : NS	US	SA	Inter-IC Sound
42	0x5002A000 0x4002A000	IPC	IPC : S IPC : NS	US	NA	Interprocessor communication
44	0x5002C000 0x4002C000	FPU	FPU : S FPU : NS	US	NA	Floating-point unit
49	0x40031000	GPIOTE	GPIOTE1	NS	NA	Non Secure GPIO tasks and events
57	0x50039000 0x40039000	KMU	KMU : S KMU : NS	SPLIT	NA	Key management unit
57	0x50039000 0x40039000	NVMC	NVMC : S NVMC : NS	SPLIT	NA	Non-volatile memory controller
58	0x5003A000 0x4003A000	VMC	VMC : S VMC : NS	US	NA	Volatile memory controller
64	0x50840000	CRYPTOCELL	CRYPTOCELL	S	NSA	CryptoCell sub-system control interface
66	0x50842500 0x40842500	GPIO	P0 : S P0 : NS	SPLIT	NA	General purpose input and output
N/A	0x00FF0000	FICR	FICR	S	NA	Factory information configuration
N/A	0x00FF8000	UICR	UICR	S	NA	User information configuration
N/A	0xE0080000	TAD	TAD	S	NA	Trace and debug control

ID	Base address	Peripheral	Instance	Secure mapping	DMA security	Description

Table 4: Instantiation table

4.2.3 Peripheral access control capabilities

Information about the peripheral access control capabilities can be found in the instantiation table.

The instantiation table has two columns containing the information about access control capabilities for a peripheral:

- Secure mapping: This column defines configuration capabilities for TrustZone®-M secure attribute.
- DMA security: This column indicates if the peripheral has DMA capabilities, and if DMA transfer can be assigned to a different security attribute than the peripheral itself.

For details on options in secure mapping column and DMA security column, see the following tables respecitvely.

Abbreviation	Description
NS	Non-secure: This peripheral is always accessible as a non-secure peripheral.
S	Secure: This peripheral is always accessible as a secure peripheral.
US	User-selectable: Non-secure or secure attribute for this peripheral is defined by the PERIPHID[0].PERM register.
SPLIT	Both non-secure and secure: The same resource is shared by both secure and non-secure code.

Table 5: Secure mapping column options

Abbreviation	Description
NA	Not applicable: Peripheral has no DMA capability.
NSA	No separate attribute: Peripheral has DMA, and DMA transfers always have the same security attribute as assigned to the peripheral.
SA	Separate attribute: Peripheral has DMA, and DMA transfers can have a different security attribute than the one assigned to the peripheral.

Table 6: DMA security column options

4.3 VMC — Volatile memory controller

The volatile memory controller (VMC) provides power control of RAM blocks.

Each of the available RAM blocks, which can contain multiple RAM sections, can be turned on or off independently in System ON mode, using the RAM[n]registers. These registers also control if a RAM block, or some of its sections, is retained in System OFF mode. See Memory chapter for more information about RAM blocks and sections.

4.3.1 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x5003A000 0x4003A000	VMC	VMC : S VMC : NS	US	NA	Volatile memory controller	

Table 7: Instances

Register	Offset	Security	Description
RAM[0].POWER	0x600		RAM0 power control register
RAM[0].POWERSET	0x604		RAM0 power control set register
RAM[0].POWERCLR	0x608		RAM0 power control clear register
RAM[1].POWER	0x610		RAM1 power control register
RAM[1].POWERSET	0x614		RAM1 power control set register
RAM[1].POWERCLR	0x618		RAM1 power control clear register
RAM[2].POWER	0x620		RAM2 power control register
RAM[2].POWERSET	0x624		RAM2 power control set register
RAM[2].POWERCLR	0x628		RAM2 power control clear register
RAM[3].POWER	0x630		RAM3 power control register
RAM[3].POWERSET	0x634		RAM3 power control set register
RAM[3].POWERCLR	0x638		RAM3 power control clear register
RAM[4].POWER	0x640		RAM4 power control register
RAM[4].POWERSET	0x644		RAM4 power control set register
RAM[4].POWERCLR	0x648		RAM4 power control clear register
RAM[5].POWER	0x650		RAM5 power control register
RAM[5].POWERSET	0x654		RAM5 power control set register
RAM[5].POWERCLR	0x658		RAM5 power control clear register
RAM[6].POWER	0x660		RAM6 power control register
RAM[6].POWERSET	0x664		RAM6 power control set register
RAM[6].POWERCLR	0x668		RAM6 power control clear register
RAM[7].POWER	0x670		RAM7 power control register
RAM[7].POWERSET	0x674		RAM7 power control set register
RAM[7].POWERCLR	0x678		RAM7 power control clear register

Table 8: Register overview

4.3.1.1 RAM[n].POWER (n=0..7)

Address offset: 0x600 + (n × 0x10)

RAMn power control register

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													H	G	F	E													D	C	B	A
Reset 0x0000FFFF	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A-D	RW	S[i]POWER (i=0..3)			Keep RAM section Si of RAM n on or off in System ON mode All RAM sections will be switched off in System OFF mode
			Off	0	Off
			On	1	On
E-H	RW	S[i]RETENTION (i=0..3)			Keep retention on RAM section Si of RAM n when RAM section is switched off
			Off	0	Off
			On	1	On

4.3.1.2 RAM[n].POWERSET (n=0..7)

Address offset: 0x604 + (n × 0x10)

RAMn power control set register

When read, this register will return the value of the POWER register.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													H	G	F	E													D	C	B	A
Reset 0x0000FFFF	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A-D	W	S[i]POWER (i=0..3)			Keep RAM section Si of RAM n on or off in System ON mode
			On	1	On
E-H	W	S[i]RETENTION (i=0..3)			Keep retention on RAM section Si of RAM n when RAM section is switched off
			On	1	On

4.3.1.3 RAM[n].POWERCLR (n=0..7)

Address offset: 0x608 + (n × 0x10)

RAMn power control clear register

When read, this register will return the value of the POWER register.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													H	G	F	E													D	C	B	A
Reset 0x0000FFFF	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A-D	W	S[i]POWER (i=0..3)			Keep RAM section Si of RAM n on or off in System ON mode
			Off	1	Off
E-H	W	S[i]RETENTION (i=0..3)			Keep retention on RAM section Si of RAM n when RAM section is switched off
			Off	1	Off

4.4 NVMC — Non-volatile memory controller

The non-volatile memory controller (NVMC) is used for writing and erasing of the internal flash memory and the user information configuration register (UICR).

The NVMC is a split security peripheral. This means that when the NVMC is configured as non-secure, only a subset of the registers is available from the non-secure code. See SPU - System protection unit on page 257 and Registers on page 31 for more details.

When the NVMC is configured to be a secure peripheral, only secure code has access.

Before a write can be performed, the NVMC must be enabled for writing in CONFIG.WEN. Similarly, before an erase can be performed, the NVMC must be enabled for erasing in CONFIG.EEN, see CONFIG on page 32. The user must make sure that writing and erasing are not enabled at the same time. Failing to do so may result in unpredictable behavior.

4.4.1 Writing to flash

When writing is enabled, in CONFIG register for secure region, or in CONFIGNS register for non-secure region, flash is written by writing a full 32-bit word to a word-aligned address in flash.

Secure code has access to both secure and non-secure regions, by using the appropriate configuration of CONFIG and CONFIGNS registers. Non-secure code, in constrast, has access to non-secure regions only. Thus, non-secure code only needs CONFIGNS.

The NVMC is only able to write '0' to erased bits in flash, that is bits set to '1'. It cannot write a bit back to '1'.

As illustrated in Memory on page 20, flash is divided into multiple pages. The same address in flash can only be written n_{WRITE} number of times before a page erase must be performed.

Only full 32-bit words can be written to flash using the NVMC interface. To write less than 32 bits to flash, write the data as a word, and set all the bits that should remain unchanged in the word to '1'. Note that the restriction about the number of writes (see above) still applies in this case.

The time it takes to write a word to flash is specified by t_{WRITE}. If CPU executes code from flash while the NVMC is writing to flash, the CPU will be stalled.

Only word-aligned writes are allowed. Byte or half-word-aligned writes will result in a bus fault.

4.4.2 Erasing a secure page in flash

When secure region erase is enabled (in CONFIG register), a flash page can be erased by writing 0xFFFFFFFF into the first 32-bit word in a flash page.

Page erase is only applicable to the code area in the flash and does not work with UICR.

After erasing a flash page, all bits in the page are set to '1'. The time it takes to erase a page is specified by $t_{ERASEPAGE}$. The CPU is stalled if the CPU executes code from the flash while the NVMC performs the erase operation.

See Partial erase of a page in flash for information on splitting the erase time in smaller chunks.

4.4.3 Erasing a non-secure page in flash

When non-secure region erase is enabled, a non-secure flash page can be erased by writing 0xFFFFFFFF into the first 32-bit word of the flash page.

Page erase is only applicable to the code area in the flash and does not work with UICR.

After erasing a flash page, all bits in the page are set to '1'. The time it takes to erase a page is specified by $t_{ERASEPAGE}$. The CPU is stalled if the CPU executes code from the flash while the NVMC performs the erase operation.

4.4.4 Writing to user information configuration registers (UICR)

User information configuration registers (UICR) are written in the same way as flash. After UICR has been written, the new UICR configuration will only take effect after a reset.

UICR is only accessible by secure code. Any write from non-secure code will be faulted. In order to lock the chip after uploading non-secure code, non-secure debugger needs to use the WRITEUICRNS register inside the NVMC in order to set APPROTECT (APPROTECT will be written to 0x00000000).

UICR can only be written n_{WRITE} number of times before an erase must be performed using ERASEALL.

The time it takes to write a word to the UICR is specified by t_{WRITE}. The CPU is stalled if the CPU executes code from the flash while the NVMC is writing to the UICR.

4.4.5 Erase all

When erase is enabled, the whole flash and UICR can be erased in one operation by using the ERASEALL register. ERASEALL will not erase the factory information configuration registers (FICR).

This functionality can be blocked by some configuration of the UICR protection bits, see the table NVMC blocking on page 30.

The time it takes to perform an ERASEALL on page 33 command is specified by $t_{ERASEALL}$. The CPU is stalled if the CPU executes code from the flash while the NVMC performs the erase operation.

4.4.6 NVMC protection mechanisms

This chapter describes the different protection mechanisms for the non-volatile memory.

4.4.6.1 NVMC blocking

UICR integrity is assured through use of multiple levels of protection. UICR protection bits can be configured to allow or block certain operations.

The table below shows the different status of UICR protection bits, and which operations are allowed or blocked.

UICR protection bit status			NVMC protection	
SECUREAPPROTECT	APPROTECT	ERASEPROTECT	CTRL-AP ERASEALL	NVMC ERASEALL
0	0	0	Available	Available
1	X	0	Available	Blocked
X	1	0	Available	Blocked
X	X	1	Blocked	Blocked

Table 9: NVMC protection (1 - Enabled, 0 - Disabled, X - Don't care)

Note: Erase can still be performed through CTRL-AP, regardless of the above settings. See CTRL-AP - Control access port on page 368 for more information.

Uploading code with secure debugging blocked

Non-secure code can program non-secure flash regions. In order to perform these operations, the NVMC has the following non-secure registers: CONFIGNS, READY and READYNEXT.

Register CONFIGNS on page 34 works as the CONFIG register but it is used only for non-secure transactions. Both page erase and writing inside the flash require a write transaction (see Erasing a secure page in flash on page 29 or Erasing a non-secure page in flash on page 29). Because of this, the SPU - System protection unit on page 257 will guarantee that the non-secure code cannot write inside a secure page, since the transaction will never reach the NVMC controller.

4.4.6.2 NVMC power failure protection

NVMC power failure protection is possible through use of power-fail comparator that is monitoring power supply.

If the power-fail comparator is enabled, and the power supply voltage is below V_{POF} threshold, the power-fail comparator will prevent the NVMC from performing erase or write operations in non-volatile memory (NVM).

If a power failure warning is present at the start of an NVM write or erase operation, the NVMC will block the operation and a bus error will be signalled. If a power failure warning occurs during an ongoing NVM write operation, the NVMC will try to finish the operation. And if the power failure warning persists, consecutive NVM write operations will be blocked by the NVMC, and a bus error will be signalled. If a power failure warning occurs during an NVM erase operation, the operation is aborted and a bus error is signalled.

4.4.7 Cache

An instruction cache (I-Cache) can be enabled for the ICODE bus in the NVMC.

See Memory map on page 22 for the location of flash.

A cache hit is an instruction fetch from the cache, and it has a 0 wait-state delay. The number of wait-states for a cache miss, where the instruction is not available in the cache and needs to be fetched from flash, depends on the processor frequency and is shown in CPU on page 19.

Enabling the cache can increase the CPU performance, and reduce power consumption by reducing the number of wait cycles and the number of flash accesses. This will depend on the cache hit rate. Cache draws current when enabled. If the reduction in average current due to reduced flash accesses is larger than the cache power requirement, the average current to execute the program code will be reduced.

When disabled, the cache does not draw current and its content is not retained.

It is possible to enable cache profiling to analyze the performance of the cache for your program using the register ICACHECNF. When profiling is enabled, registers IHIT and IMISS are incremented for every instruction cache hit or miss respectively.

4.4.8 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50039000 0x40039000	NVMC	NVMC : S NVMC : NS	SPLIT	NA	Non-volatile memory controller	

Table 10: Instances

Register	Offset	Security	Description
READY	0x400	NS	Ready flag
READYNEXT	0x408	NS	Ready flag
CONFIG	0x504	S	Configuration register
ERASEALL	0x50C	S	Register for erasing all non-volatile user memory
ERASEPAGEPARTIALCFG	0x51C	S	Register for partial erase configuration
ICACHECNF	0x540	S	I-code cache configuration register
IHIT	0x548	S	I-code cache hit counter
IMISS	0x54C	S	I-code cache miss counter
CONFIGNS	0x584	NS	
WRITEUICRNS	0x588	NS	Non-secure APPROTECT enable register
FORCEONNVM	0x700	S	Force on all NVM supplies. Also see the internal section in the NVMC chapter.

Register	Offset	Security	Description
FORCEOFFNVM	0x728	S	Force off NVM supply. Also see the internal section in the NVMC chapter.

Table 11: Register overview

4.4.8.1 READY

Address offset: 0x400

Ready flag

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000001	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	READY			NVMC is ready or busy
			Busy	0	NVMC is busy (on-going write or erase operation)
			Ready	1	NVMC is ready

4.4.8.2 READYNEXT

Address offset: 0x408

Ready flag

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000001	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	READYNEXT			NVMC can accept a new write operation
			Busy	0	NVMC cannot accept any write operation
			Ready	1	NVMC is ready

4.4.8.3 CONFIG

Address offset: 0x504

Configuration register

This register is one hot

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	WEN			Program memory access mode. It is strongly recommended to only activate erase and write modes when they are actively used. Enabling write or erase will invalidate the cache and keep it invalidated.
			Ren	0	Read only access
			Wen	1	Write enabled
			Een	2	Erase enabled
			PEen	4	Partial erase enabled

4.4.8.4 ERASEALL

Address offset: 0x50C

Register for erasing all non-volatile user memory

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	ERASEALL			Erase all non-volatile memory including UICR registers. Note that erasing must be enabled by setting CONFIG.WEN = Een before the non-volatile memory can be erased.
			NoOperation	0	No operation
			Erase	1	Start chip erase

4.4.8.5 ERASEPAGEPARTIALCFG

Address offset: 0x51C

Register for partial erase configuration

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										A	A	A	A	A	A	A
Reset 0x0000000A	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	DURATION			Duration of the partial erase in milliseconds The user must ensure that the total erase time is long enough for a complete erase of the flash page

4.4.8.6 ICACHECNF

Address offset: 0x540

I-code cache configuration register

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																								B								A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CACHEEN			Cache enable
			Disabled	0	Disable cache. Invalidates all cache entries.
			Enabled	1	Enable cache
B	RW	CACHEPROFEN			Cache profiling enable
			Disabled	0	Disable cache profiling
			Enabled	1	Enable cache profiling

4.4.8.7 IHIT

Address offset: 0x548

I-code cache hit counter

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	HITS			Number of cache hits

4.4.8.8 IMISS

Address offset: 0x54C

I-code cache miss counter

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	MISSES			Number of cache misses

4.4.8.9 CONFIGNS

Address offset: 0x584

This register is one hot

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	WEN			Program memory access mode. It is strongly recommended to only activate erase and write modes when they are actively used. Enabling write or erase will invalidate the cache and keep it invalidated.
			Ren	0	Read only access
			Wen	1	Write enabled
			Een	2	Erase enabled

4.4.8.10 WRITEUICRNS

Address offset: 0x588

Non-secure APPROTECT enable register

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B				A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	SET			Allow non-secure code to set APPROTECT
			Set	1	Set value
B	W	KEY			Key to write in order to validate the write operation
			Keyvalid	0xAFBE5A7	Key value

4.4.8.11 FORCEONNVM

Address offset: 0x700

Force on all NVM supplies. Also see the internal section in the NVMC chapter.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	FORCEONNVM			Force on all NVM supplies. Also see the internal section in the NVMC chapter.
			DoNotForceOn	0	Do not force on NVM supply
			ForceOn	1	Force on NVM supply

4.4.8.12 FORCEOFFNVM

Address offset: 0x728

Force off NVM supply. Also see the internal section in the NVMC chapter.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C							B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	FORCEOFFNVM0			Force off NVM supply 0. Also see the internal section in the NVMC chapter.
			DoNotForceOff	0	Do not force off supply
			ForceOff	1	Force off supply
B	RW	FORCEOFFNVM1			Force off NVM supply 1. Also see the internal section in the NVMC chapter.
			DoNotForceOff	0	Do not force off supply
			ForceOff	1	Force off supply
C	RW	KEY			KEY
			EnableWrite	0xACCE55	Must be written in order to write to bits 0-7. Any other value will ignore writes to this register. Read as zero.

4.4.9 Electrical specification

4.4.9.1 Flash programming

Symbol	**Description**	**Min.**	**Typ.**	**Max.**	**Units**
n_{WRITE}	Number of times a 32-bit word can be written before erase			2	
$n_{ENDURANCE}$	Erase cycles per page	10,000			
t_{WRITE}	Time to write one 32-bit word			43	µs
$t_{ERASEPAGE}$	Time to erase one page			87	ms
$t_{ERASEALL}$	Time to erase all flash			173	ms
$t_{ERASEPAGEPARTIAL,setup}$	Setup time for one partial erase			1.08	ms

4.4.9.2 Cache size

Symbol	**Description**	**Min.**	**Typ.**	**Max.**	**Units**
$Size_{ICODE}$	I-Code cache size		2048		Bytes

4.5 FICR — Factory information configuration registers

Factory information configuration registers (FICR) are pre-programmed in factory and cannot be erased by the user. These registers contain chip-specific information and configuration.

4.5.1 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x00FF0000	FICR	FICR	S	NA	Factory information configuration	

Table 12: Instances

Register	Offset	Security	Description
INFO.DEVICEID[0]	0x204		Device identifier
INFO.DEVICEID[1]	0x208		Device identifier
INFO.PART	0x20C		Part code
INFO.VARIANT	0x210		Part Variant, Hardware version and Production configuration
INFO.PACKAGE	0x214		Package option
INFO.RAM	0x218		RAM variant
INFO.FLASH	0x21C		Flash variant
INFO.CODEPAGESIZE	0x220		Code memory page size
INFO.CODESIZE	0x224		Code memory size
INFO.DEVICETYPE	0x228		Device type
TRIMCNF[n].ADDR	0x300		Address
TRIMCNF[n].DATA	0x304		Data
TRNG90B.BYTES	0xC00		Amount of bytes for the required entropy bits
TRNG90B.RCCUTOFF	0xC04		Repetition counter cutoff
TRNG90B.APCUTOFF	0xC08		Adaptive proportion cutoff
TRNG90B.STARTUP	0xC0C		Amount of bytes for the startup tests
TRNG90B.ROSC1	0xC10		Sample count for ring oscillator 1
TRNG90B.ROSC2	0xC14		Sample count for ring oscillator 2
TRNG90B.ROSC3	0xC18		Sample count for ring oscillator 3
TRNG90B.ROSC4	0xC1C		Sample count for ring oscillator 4

Table 13: Register overview

4.5.1.1 INFO.DEVICEID[n] (n=0..1)

Address offset: 0x204 + (n × 0x4)

Device identifier

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	DEVICEID			64 bit unique device identifier DEVICEID[0] contains the least significant bits of the device identifier. DEVICEID[1] contains the most significant bits of the device identifier.

4.5.1.2 INFO.PART

Address offset: 0x20C

Part code

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00009160	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	PART			Part code
			N9160	0x9160	nRF9160

4.5.1.3 INFO.VARIANT

Address offset: 0x210

Part Variant, Hardware version and Production configuration

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x0FFFFFFF	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	VARIANT			Part Variant, Hardware version and Production configuration, encoded as ASCII
			AAAA	0x41414141	AAAA
			AAA0	0x41414130	AAA0

4.5.1.4 INFO.PACKAGE

Address offset: 0x214

Package option

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00002000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	PACKAGE			Package option
			CC	0x2000	CCxx - 236 ball wlCSP

4.5.1.5 INFO.RAM

Address offset: 0x218

RAM variant

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000100	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	RAM			RAM variant
			K256	0x100	256 kByte RAM
			Unspecified	0xFFFFFFFF	Unspecified

4.5.1.6 INFO.FLASH

Address offset: 0x21C

Flash variant

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000400	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	FLASH			Flash variant
			K1024	0x400	1 MByte FLASH

4.5.1.7 INFO.CODEPAGESIZE

Address offset: 0x220

Code memory page size

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00001000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	CODEPAGESIZE			Code memory page size

4.5.1.8 INFO.CODESIZE

Address offset: 0x224

Code memory size

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000100	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	CODESIZE			Code memory size in number of pages Total code space is: CODEPAGESIZE * CODESIZE

4.5.1.9 INFO.DEVICETYPE

Address offset: 0x228

Device type

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	DEVICETYPE			Device type
			Die	0x0000000	Device is an physical DIE
			FPGA	0xFFFFFFFF	Device is an FPGA

4.5.1.10 TRIMCNF[n].ADDR (n=0..255)

Address offset: 0x300 + (n × 0x8)

Address

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	Address			Address

4.5.1.11 TRIMCNF[n].DATA (n=0..255)

Address offset: 0x304 + (n × 0x8)

Data

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	Data			Data

4.5.1.12 TRNG90B.BYTES

Address offset: 0xC00

Amount of bytes for the required entropy bits

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	BYTES			Amount of bytes for the required entropy bits

4.5.1.13 TRNG90B.RCCUTOFF

Address offset: 0xC04

Repetition counter cutoff

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	RCCUTOFF			Repetition counter cutoff

4.5.1.14 TRNG90B.APCUTOFF

Address offset: 0xC08

Adaptive proportion cutoff

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	APCUTOFF			Adaptive proportion cutoff

4.5.1.15 TRNG90B.STARTUP

Address offset: 0xC0C

Amount of bytes for the startup tests

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000210	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	STARTUP			Amount of bytes for the startup tests

4.5.1.16 TRNG90B.ROSC1

Address offset: 0xC10

Sample count for ring oscillator 1

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	ROSC1			Sample count for ring oscillator 1

4.5.1.17 TRNG90B.ROSC2

Address offset: 0xC14

Sample count for ring oscillator 2

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	ROSC2			Sample count for ring oscillator 2

4.5.1.18 TRNG90B.ROSC3

Address offset: 0xC18

Sample count for ring oscillator 3

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	ROSC3			Sample count for ring oscillator 3

4.5.1.19 TRNG90B.ROSC4

Address offset: 0xC1C

Sample count for ring oscillator 4

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	ROSC4			Sample count for ring oscillator 4

4.6 UICR — User information configuration registers

The user information configuration registers (UICRs) are non-volatile memory (NVM) registers for configuring user specific settings.

For information on writing UICR registers, see the NVMC — Non-volatile memory controller on page 28 and Memory on page 20 chapters.

4.6.1 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x00FF8000	UICR	UICR	S	NA	User information configuration	

Table 14: Instances

Register	Offset	Security	Description
APPROTECT	0x000		Access port protection
UNUSED0	0x004		Reserved
UNUSED1	0x008		Reserved
UNUSED2	0x00C		Reserved
UNUSED3	0x010		Reserved
XOSC32M	0x014		Oscillator control
HFXOSRC	0x01C		HFXO clock source selection
HFXOCNT	0x020		HFXO startup counter
SECUREAPPROTECT	0x02C		Secure access port protection
ERASEPROTECT	0x030		Erase protection
OTP[n]	0x108		OTP bits [31+n*32:0+n*32].
KEYSLOT.CONFIG[n].DEST	0x400		Destination address where content of the key value registers (KEYSLOT.KEYn.VALUE[0-3]) will be pushed by KMU. Note that this address MUST match that of a peripherals APB mapped write-only key registers, else the KMU can push this key value into an address range which the CPU can potentially read!
KEYSLOT.CONFIG[n].PERM	0x404		Define permissions for the key slot with ID=n+1. Bits 0-15 and 16-31 can only be written once.
KEYSLOT.KEY[n].VALUE[0]	0x800		Define bits [31:0] of value assigned to KMU key slot ID=n+1
KEYSLOT.KEY[n].VALUE[1]	0x804		Define bits [63:32] of value assigned to KMU key slot ID=n+1
KEYSLOT.KEY[n].VALUE[2]	0x808		Define bits [95:64] of value assigned to KMU key slot ID=n+1
KEYSLOT.KEY[n].VALUE[3]	0x80C		Define bits [127:96] of value assigned to KMU key slot ID=n+1

Table 15: Register overview

4.6.1.1 APPROTECT

Address offset: 0x000

Access port protection

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	PALL			Blocks debugger read/write access to all CPU registers and memory mapped addresses
			Unprotected	0xFFFFFFFF	Unprotected
			Protected	0x00000000	Protected

4.6.1.2 XOSC32M

Address offset: 0x014

Oscillator control

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																											A	A	A	A	A	A
Reset 0xFFFFFFCF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	CTRL			Pierce current DAC control signals

4.6.1.3 HFXOSRC

Address offset: 0x01C

HFXO clock source selection

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	HFXOSRC			HFXO clock source selection
			XTAL	1	32 MHz crystal oscillator
			TCXO	0	32 MHz temperature compensated crystal oscillator (TCXO)

4.6.1.4 HFXOCNT

Address offset: 0x020

HFXO startup counter

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	HFXOCNT			HFXO startup counter. Total debounce time = HFXOCNT*64 us + 0.5 us
			MinDebounceTime	0	Min debounce time = (0*64 us + 0.5 us)
			MaxDebounceTime	255	Max debounce time = (255*64 us + 0.5 us)

4.6.1.5 SECUREAPPROTECT

Address offset: 0x02C

Secure access port protection

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	PALL			Blocks debugger read/write access to all secure CPU registers and secure memory mapped addresses
			Unprotected	0xFFFFFFFF	Unprotected
			Protected	0x00000000	Protected

4.6.1.6 ERASEPROTECT

Address offset: 0x030

Erase protection

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	PALL			Blocks NVMC ERASEALL and CTRLAP ERASEALL functionality
			Unprotected	0xFFFFFFFF	Unprotected
			Protected	0x00000000	Protected

4.6.1.7 OTP[n] (n=0..189)

Address offset: 0x108 + (n × 0x4)

OTP bits [31+n*32:0+n*32].

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	OTP			Bits [31+n*32:0+n*32] of OTP region

4.6.1.8 KEYSLOT.CONFIG[n].DEST (n=0..127)

Address offset: 0x400 + (n × 0x8)

Destination address where content of the key value registers (KEYSLOT.KEYn.VALUE[0-3]) will be pushed by KMU. Note that this address MUST match that of a peripherals APB mapped write-only key registers, else the KMU can push this key value into an address range which the CPU can potentially read!

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	DEST			Secure APB destination address

4.6.1.9 KEYSLOT.CONFIG[n].PERM (n=0..127)

Address offset: 0x404 + (n × 0x8)

Define permissions for the key slot with ID=n+1. Bits 0-15 and 16-31 can only be written once.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																D														C	B	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	WRITE			Write permission for key slot
			Disabled	0	Disable write to the key value registers
			Enabled	1	Enable write to the key value registers
B	RW	READ			Read permission for key slot
			Disabled	0	Disable read from key value registers
			Enabled	1	Enable read from key value registers
C	RW	PUSH			Push permission for key slot
			Disabled	0	Disable pushing of key value registers over secure APB, but can be read if field READ is Enabled
			Enabled	1	Enable pushing of key value registers over secure APB. Register KEYSLOT.CONFIGn.DEST must contain a valid destination address!
D	RW	STATE			Revocation state for the key slot Note that it is not possible to undo a key revocation by writing the value '1' to this field
			Revoked	0	Key value registers can no longer be read or pushed
			Active	1	Key value registers are readable (if enabled) and can be pushed (if enabled)

4.6.1.10 KEYSLOT.KEY[n].VALUE[o] (n=0..127) (o=0..3)

Address offset: 0x800 + (n × 0x10) + (o × 0x4)

Define bits [31+o*32:0+o*32] of value assigned to KMU key slot ID=n+1

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	VALUE			Define bits [31+o*32:0+o*32] of value assigned to KMU key slot ID=n+1

4.7 EasyDMA

EasyDMA is a module implemented by some peripherals to gain direct access to Data RAM.

EasyDMA is an AHB bus master similar to CPU and is connected to the AHB multilayer interconnect for direct access to Data RAM. EasyDMA is not able to access flash.

A peripheral can implement multiple EasyDMA instances to provide dedicated channels. For example, for reading and writing of data between the peripheral and RAM. This concept is illustrated in EasyDMA example on page 45.

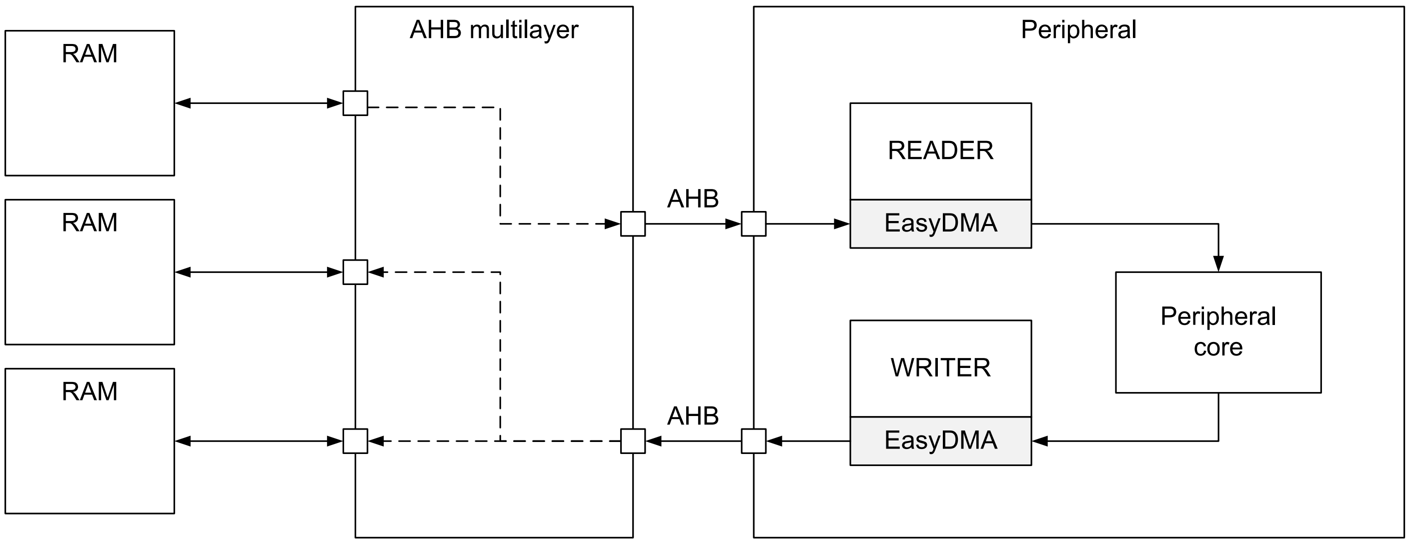


Figure 5: EasyDMA example

An EasyDMA channel is usually implemented like illustrated by the code below, but some variations may occur:

```
READERBUFFER_SIZE 5
WRITERBUFFER_SIZE 6

uint8_t readerBuffer[READERBUFFER_SIZE]  __at__ 0x20000000;
uint8_t writerBuffer[WRITERBUFFER_SIZE]  __at__ 0x20000005;

// Configuring the READER channel
MYPERIPHERAL->READER.MAXCNT = READERBUFFER_SIZE;
MYPERIPHERAL->READER.PTR = &readerBuffer;

// Configure the WRITER channel
MYPERIPHERAL->WRITER.MAXCNT = WRITEERBUFFER_SIZE;
MYPERIPHERAL->WRITER.PTR = &writerBuffer;
```

This example shows a peripheral called MYPERIPHERAL that implements two EasyDMA channels - one for reading called READER, and one for writing called WRITER. When the peripheral is started, it is assumed that the peripheral will:

- Read 5 bytes from the readerBuffer located in RAM at address 0x20000000.
- Process the data.
- Write no more than 6 bytes back to the writerBuffer located in RAM at address 0x20000005.

The memory layout of these buffers is illustrated in EasyDMA memory layout on page 46.

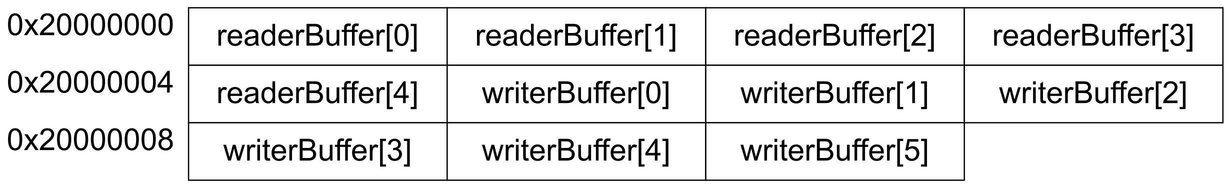


Figure 6: EasyDMA memory layout

The WRITER.MAXCNT register should not be specified larger than the actual size of the buffer (writerBuffer). Otherwise, the channel would overflow the writerBuffer.

Once an EasyDMA transfer is completed, the AMOUNT register can be read by the CPU to see how many bytes were transferred. For example, CPU can read MYPERIPHERAL->WRITER.AMOUNT register to see how many bytes WRITER wrote to RAM.

4.7.1 EasyDMA array list

EasyDMA is able to operate in a mode called array list.

The array list does not provide a mechanism to explicitly specify where the next item in the list is located. Instead, it assumes that the list is organized as a linear array where items are located one after the other in RAM.

The EasyDMA array list can be implemented by using the data structure ArrayList_type as illustrated in the code example below:

```
#define BUFFER_SIZE  4

typedef struct ArrayList
{
  uint8_t buffer[BUFFER_SIZE];
} ArrayList_type;

ArrayList_type ReaderList[3];

READER.MAXCNT = BUFFER_SIZE;
READER.PTR = &ReaderList;
```

The data structure only includes a buffer with size equal to the size of READER.MAXCNT register. EasyDMA uses the READER.MAXCNT register to determine when the buffer is full.

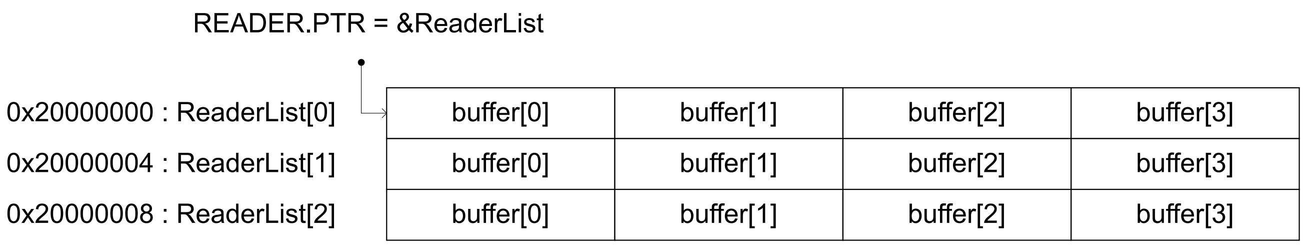

READER.PTR = &ReaderList
0x20000000 : ReaderList[0]
0x20000004 : ReaderList[1]
0x20000008 : ReaderList[2]
buffer[0]
buffer[1]
buffer[2]
buffer[3]

Figure 7: EasyDMA array list

4.8 AHB multilayer interconnect

On the AHB multilayer interconnect, the application CPU and all EasyDMA instances are AHB bus masters while RAM, cache and peripherals are AHB slaves. External MCU subsystems can be seen both as master and slave on the AHB multilayer interconnect.

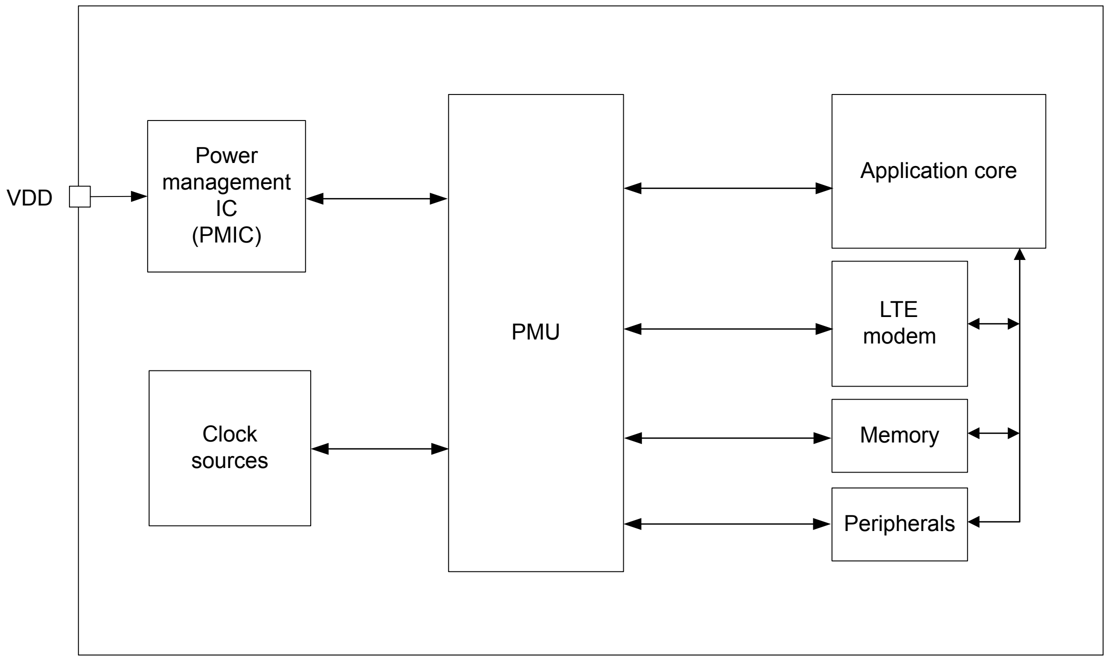
Multiple AHB masters can access slave resources within the AHB multilayer interconnect as illustrated in Memory on page 20. Access rights to each of the AHB slaves are resolved using the natural priority of the different bus masters in the system.

5 Power and clock management

5.1 Functional description

The power and clock management system automatically ensures maximum power efficiency.

The core of the power and clock management is the power management unit (PMU) illustrated in the image below.



The PMU automatically tracks the power and clock resources required by the different components in the system. It then starts/stops and chooses operation modes in supply regulators and clock sources, without user interaction, to achieve the lowest power consumption possible.

5.1.1 Power management

The power management unit (PMU) handles two system-wide power modes - System ON and System OFF.

Internal blocks of the device are automatically powered by the PMU as they are required by the application.

5.1.1.1 System ON mode

System ON is the power mode entered after a power-on reset.

While in System ON, the system can reside in one of two sub modes:

- Low power
- Constant latency

The low power mode is default after power-on reset.

In low power mode, whenever no application or wireless activity takes place, function blocks like the application CPU, LTE modem and all peripherals are in IDLE state. That particular state is referred to as System ON IDLE. In this state, all function blocks retain their state and configuration, so they are ready to become active once configured by the CPU.

If any application or modem activity occurs, the system leaves the System ON IDLE state. Once a given activity in a function block is completed, the system automatically returns to IDLE, retaining its configuration.

As long as the system resides in low power mode, the PMU ensures that the appropriate regulators and clock sources are started or stopped based on the needs of the function blocks active at any given time.

This automatic power management can be overridden by switching to constant latency mode. In this mode, the CPU wakeup latency and the PPI task response are constant and kept at a minimum. This is secured by keeping a set of base resources that are always enabled. The advantage of having a constant and predictable latency will be at the cost of having significantly increased power consumption compared to the low power mode. The constant latency mode is enabled by triggering the CONSTLAT task (TASKS_CONSTLAT on page 59).

While the system is in constant latency mode, the low power mode can be enabled by triggering LOWPWR task (TASKS_LOWPWR on page 59).

To reduce power consumption while in System ON IDLE, RAM blocks can be turned off in System ON mode while enabling the retention of these RAM blocks in RAM[n].POWER registers in VMC. RAM[n].POWER are retained registers, see Reset behavior on page 55. Note that these registers are usually overwritten by the startup code provided with the nRF application examples.

5.1.1.2 System OFF mode

System OFF is the deepest power saving mode the system can enter.

In this mode, the core system functionality is powered down and ongoing tasks terminated, and only the reset and the wakeup functions are available and responsive.

The device is put into System OFF mode using the REGULATORS register interface. When in System OFF mode, one of the following signals/actions will wake up the device:

1. DETECT signal, generated by the GPIO peripheral
2. RESET
3. start of debug session

When the device wakes up from System OFF mode, a system reset is performed.

One or more RAM blocks can be retained in System OFF mode depending on the settings in the RAM[n].POWER registers in VMC. RAM[n].POWER are retained registers, see Reset behavior on page 55. Note that these registers are usually overwritten by the startup code provided with the nRF application examples.

Before entering System OFF mode, the user must make sure that all on-going EasyDMA transactions have completed. This can be accomplished by making sure that EasyDMA enabled peripherals have stopped and END events from them received. The LTE modem also needs to be stopped, by issuing a command through the modem API, before entering System OFF mode. Once the command is issued, one should wait for the modem to respond that it actually has stopped, as there may be a delay until modem is disconnected from the network.

5.1.1.2.1 Emulated System OFF mode

If the device is in debug interface mode, System OFF will be emulated to secure that all required resources needed for debugging are available during System OFF.

See Overview on page 365 chapter for more information. Required resources needed for debugging include the following key components: Overview on page 365, CLOCK — Clock control on page 64, POWER — Power control on page 58, NVMC — Non-volatile memory controller on page 28, CPU on page 19, flash, and RAM. Since the CPU is kept on in emulated System OFF mode, it is required to add an infinite loop directly after entering System OFF, to prevent the CPU from executing code that normally should not be executed.

5.1.2 Power supply

The device has a single main power supply VDD, and the internal components are powered by integrated voltage regulators. The PMU manages these regulators automatically, no voltage regulator control needs to be included in application firmware.

5.1.2.1 General purpose I/O supply

The input/output (I/O) drivers of P0.00 - P0.31 pins are supplied independently of VDD through VDD_GPIO. This enables easy match to signal voltage levels in the printed circuit board design.

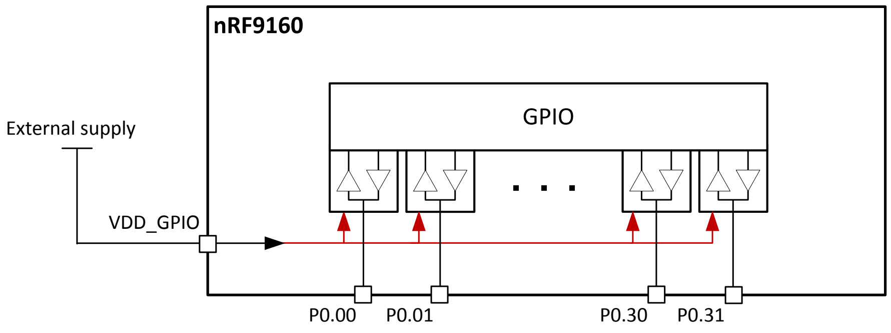


Figure 8: GPIO supply input (VDD_GPIO)

The I/Os are supplied via VDD_GPIO pin as shown in figure above. VDD_GPIO pin supports voltage levels within range given in table Recommended operating conditions on page 386

5.1.3 Power supply monitoring

Power monitor solutions are available in the device, in order to survey the VDD (battery voltage).

5.1.3.1 Power supply supervisor

The power supply supervisor enables monitoring of the connected power supply.

Two functionalities are implemented:

- Power-on reset (POR): Generates a reset when the supply is applied to the device, and ensures that the device starts up in a known state
- Brownout reset (BOR): Generates a reset when the supply drops below the minimum voltage required for safe operations

Two BOR levels are used:

- V_{BOROFF}, used in System OFF
- V_{BORON}, used in System ON

The power supply supervisor is illustrated in the image below.

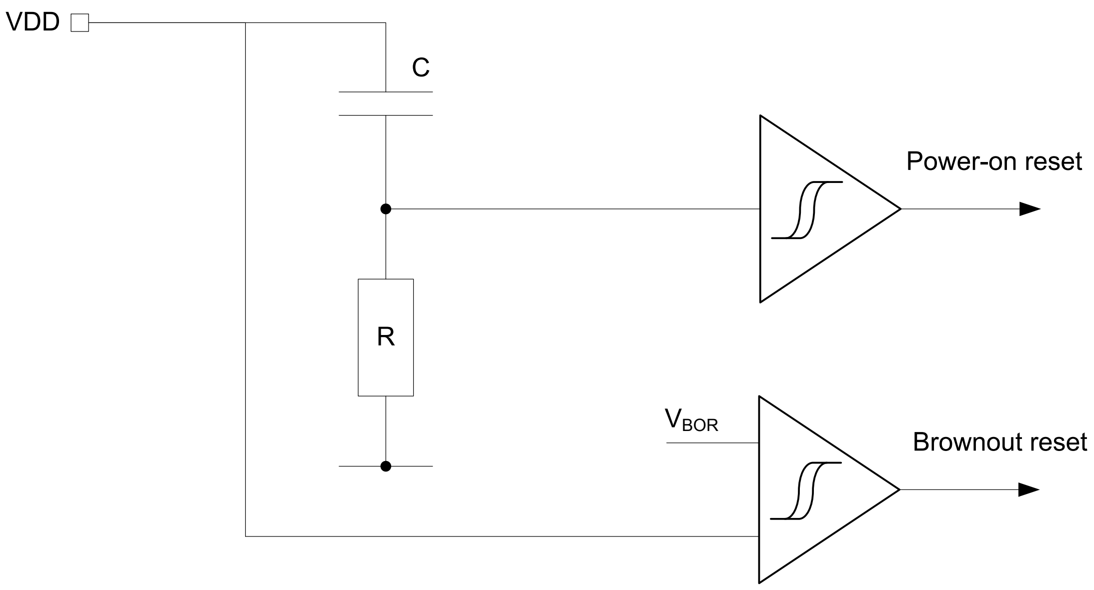


Figure 9: Power supply supervisor

5.1.3.2 Battery monitoring on VDD

A battery voltage (VDD) monitoring capability is provided via a modem API

Note: For details regarding the modem API, please refer to *nRF Connect SDK* document and *nRF91 AT Commands, Command Reference Guide* document.

5.1.3.3 Electrical specification

5.1.4 Clock management

The clock control system can source the system clocks from a range of high and low frequency oscillators, and distribute them to modules based upon a module's individual requirements. Clock generation and distribution is handled automatically by PMU to optimize current consumption.

Listed here are the available clock signal sources:

- 64 MHz oscillator (HFINT)
- 64 MHz high accuracy oscillator (HFXO)
- 32.768 kHz RC oscillator (LFRC)
- 32.768 kHz high accuracy oscillator (LFXO)

The clock and oscillator resources are configured and controlled via the CLOCK peripheral as illustrated below.

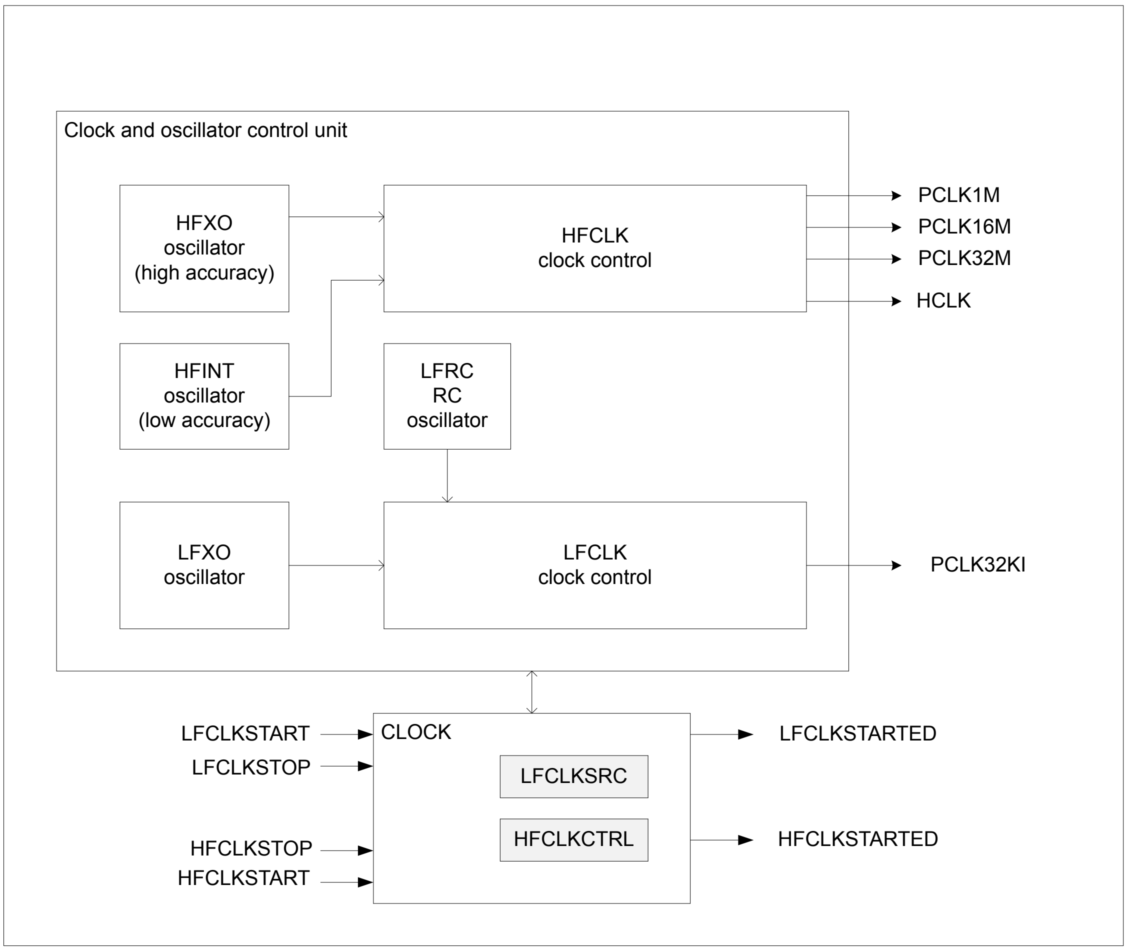


Figure 10: Clock and oscillator setup

5.1.4.1 HFCLK clock controller

The HFCLK clock controller provides several clocks in the system.

These are as follows:

- HCLK: 64 MHz CPU clock
- PCLK1M: 1 MHz peripheral clock
- PCLK16M: 16 MHz peripheral clock
- PCLK32M: 32 MHz peripheral clock

The HFCLK controller uses the following high frequency clock (HFCLK) sources:

- 64 MHz oscillator (HFINT)
- 64 MHz high accuracy oscillator (HFXO)

For illustration, see Clock and oscillator setup on page 52.

The HFCLK controller will automatically provide the clock(s) requested by the system. If the system does not request any clocks from the HFCLK controller, the controller will switch off all its clock sources and enter a power saving mode.

The HFINT source will be used when HFCLK is requested and HFXO has not been started.

The HFXO is started by triggering the HFCLKSTART task and stopped using the HFCLKSTOP task. A HFCLKSTARTED event will be generated when the HFXO has started and its frequency is stable.

5.1.4.2 LFCLK clock controller

The system supports several low frequency clock sources.

As illustrated in Clock and oscillator setup on page 52, the system supports the following low frequency clock sources:

- LFRC: 32.768 kHz RC oscillator
- LFXO: 32.768 kHz high accuracy oscillator

The LFCLK clock controller and all LFCLK clock sources are always switched off when in System OFF mode.

The LFCLK clock is started by first selecting the preferred clock source in the LFCLKSRC on page 71 register and then triggering the LFCLKSTART task. LFXO is recommended as the LFCLK clock source.

Note: The LTE modem requires using LFXO as the LFCLK source.

Switching between LFCLK clock sources can be done without stopping the LFCLK clock. A LFCLK clock source which is running prior to triggering the LFCLKSTART task will continue to run until the selected clock source has been available. After that the clock sources will be switched. Switching between clock sources will never introduce a glitch but it will stretch a clock pulse by 0.5 to 1.0 clock cycle (i.e. will delay rising edge by 0.5 to 1.0 clock cycle).

Note: If the watchdog timer (WDT) is running, the default LFCLK clock source (LFRC - see LFCLKSRC on page 71) is started automatically (LFCLKSTART task doesn't have to be triggered).

A LFCLKSTARTED event will be generated when the selected LFCLK clock source has started.

A LFCLKSTOP task will stop global requesting of the LFCLK clock. However, if any system component (e.g. WDT, modem) requires the LFCLK, the clock won't be stopped. The LFCLKSTOP task should only be triggered after the STATE field in the LFCLKSTAT register indicates a LFCLK running-state.

5.1.4.2.1 32.768 kHz RC oscillator (LFRC)

The default source of the low frequency clock (LFCLK) is the 32.768 kHz RC oscillator (LFRC).

The LFRC frequency will be affected by variation in temperature.

5.1.4.3 Electrical specification

5.1.4.3.1 64 MHz internal oscillator (HFINT)

Symbol	Description	Min.	Typ.	Max.	Units
f_{NOM_HFINT}	Nominal output frequency		64		MHz
f_{TOL_HFINT}	Frequency tolerance		+-1	+-5	%
t_{START_HFINT}	Startup time		3.2		µs

5.1.4.3.2 64 MHz high accuracy oscillator (HFXO)

Symbol	Description	Min.	Typ.	Max.	Units
f_{NOM_HFXO}	Nominal output frequency		64		MHz
f_{TOL_HFXO}	Frequency tolerance		+-1		ppm
t_{START_HFXO}	Startup time		TBA		ms

5.1.4.3.3 32.768 kHz high accuracy oscillator (LFXO)

Symbol	Description	Min.	Typ.	Max.	Units
f_{NOM_LFXO}	Frequency		32.768		kHz
f_{TOL_LFXO}	Frequency tolerance		+-20		ppm
t_{START_LFXO}	Startup time		TBA		s

5.1.4.3.4 32.768 kHz RC oscillator (LFRC)

Symbol	Description	Min.	Typ.	Max.	Units
f_{NOM_LFRC}	Nominal frequency		32.768		kHz
f_{TOL_LFRC}	Frequency tolerance		+-2		%
t_{START_LFRC}	Startup time		600		µs

5.1.5 Reset

There are multiple reset sources that may trigger a reset of the system. After a reset the CPU can query the RESETREAS (reset reason register) to find out which source generated the reset.

5.1.5.1 Power-on reset

The power-on reset generator initializes the system at power-on. The system is held in reset state until the supply has reached the minimum operating voltage and the internal voltage regulators have started.

5.1.5.2 Pin reset

A pin reset is generated when the physical reset pin (nRESET) on the device is pulled low.

To ensure that reset is issued correctly, the reset pin should be held low for time given in .

nRESET pin has an always-on internal pull-up resistor connected to VDD. The value of the pull-up resistor is given in .

5.1.5.3 Wakeup from System OFF mode reset

The device is reset when it wakes up from System OFF mode.

The Debug access port is not reset following a wake up from System OFF mode if the device is in debug interface mode, see Overview on page 365 chapter for more information.

5.1.5.4 Soft reset

A soft reset is generated when the SYSRESETREQ bit of the application interrupt and reset control register (AIRCR register) in the ARM® core is set.

5.1.5.5 Watchdog reset

A watchdog reset is generated when the watchdog timer (WDT) times out.

See WDT — Watchdog timer on page 352 chapter for more information.

5.1.5.6 Brownout reset

The brownout reset generator puts the system in reset state if the supply voltage drops below the brownout reset threshold.

5.1.5.7 Retained registers

A retained register is a register that will retain its value in System OFF mode, and through a reset depending on reset source. See individual peripheral chapters for information of which registers are retained for the different peripherals.

5.1.5.8 Reset behavior

Reset behavior depends on the reset source.

The reset behavior is summarized in the table below.

Reset source	Reset target							
	CPU	Modem	Debug[2]	SWJ-DP	Not retained RAM[3]	Retained RAM[3]	WDT	RESETREAS
CPU lockup [4]	x	x						
Soft reset	x	x						
Wakeup from System OFF mode reset	x	x	x [5]		x		x	
Watchdog reset [6]	x	x	x		x		x	
Pin reset	x	x	x	x	x		x	
Brownout reset	x	x	x	x	x	x	x	x
Power-on reset	x	x	x	x	x	x	x	

Table 16: Reset behavior for the main components

Note: The RAM is never reset but its content may be corrupted after reset in the cases given in the table above.

Reset source	Reset target					
	Regular peripheral registers	GPIO, SPU	NVMC WAITSTATENUM	NVMC IFCREADDELAY	REGULATORS, OSCILLATORS	POWER.GPREGRET
CPU lockup[4]	x	x	x			
Soft reset	x	x	x			
Wakeup from System OFF mode reset	x		x			
Watchdog reset[6]	x	x	x		x	
Pin reset	x	x	x		x	
Brownout reset	x	x	x	x	x	x
Power-on reset	x	x	x	x	x	x

Table 17: Reset behavior for the retained registers

5.1.5.9 Electrical specification

5.1.5.9.1 Pin reset

Symbol	Description	Min.	Typ.	Max.	Units
$t_{HOLDRESET}$	Hold time for reset pin when doing a pin reset	..	..	..	µs
$R_{PULL-UP}$	Value of the internal pull-up resistor	..	..	..	kΩ

[2] All debug components excluding SWJ-DP. See Overview on page 365 chapter for more information about the different debug components in the system.

[3] RAM can be configured to be retained using registers in VMC — Volatile memory controller on page 26

[4] Reset from CPU lockup is disabled if the device is in debug interface mode. CPU lockup is not possible in System OFF.

[5] The debug components will not be reset if the device is in debug interface mode.

[6] Watchdog reset is not available in System OFF.

5.2 Current consumption

As the system is being constantly tuned by the PMU described in Functional description on page 48, estimating the current consumption of an application can be challenging if the designer is not able to perform measurements directly on the hardware. To facilitate the estimation process, a set of current consumption scenarios are provided to show the typical current drawn from the VDD supply.

Each scenario specifies a set of operations and conditions applying to the given scenario. Current consumption scenarios, common conditions on page 56 shows a set of common conditions used in all scenarios, unless otherwise is stated in the description of a given scenario. Current consumption scenarios, common conditions on page 56 describes the conditions used for the modem current consumption specifications. All scenarios are listed in Electrical specification on page 56

Condition	Value
Supply	3.7 V
Temperature	25 °C
CPU	WFI (wait for interrupt)/WFE (wait for event) sleep
Peripherals	All idle
Clock	Not running
RAM	No retention
Cache enabled	Yes

Table 18: Current consumption scenarios, common conditions

Condition
Cat-M1 HD FDD mode
Ideal channel, no errors in DL/UL communication
Network response times at minimum
UICC current consumption excluded
Output power at antenna port, single-ended 50 Ω

Table 19: Current consumption scenarios, common conditions

5.2.1 Electrical specification

5.2.1.1 Sleep

Symbol	Description	Min.	Typ.	Max.	Units
$I_{MCUOFF0}$	MCU off, modem off, no RAM retention, wake on GPIO and reset		1.4		µA
I_{MCUON0}	MCU on IDLE, modem off, RTC off		1.8		µA
I_{MCUON1}	MCU on IDLE, modem off, RTC on		2.35		µA

5.2.1.2 Application CPU active current consumption

Symbol	Description	Min.	Typ.	Max.	Units
I_{CPU0_FLASH}	CPU running CoreMark @64 MHz from flash, clock = HFXO, cache enabled		2.88		mA
$I_{COREMARK\_PER\_MA\_FL/}$	CoreMark per mA, executing from flash, CoreMark=243		84		CoreMark/ mA
I_{CPU0_RAM}	CPU running CoreMark @64 MHz from RAM, clock = HFXO, cache enabled		2.32		mA
$I_{COREMARK\_PER\_MA\_RA}$	CoreMark per mA, executing from RAM, CoreMark=235		101		CoreMark/ mA

5.2.1.3 I2S

Symbol	Description	Min.	Typ.	Max.	Units
I_{I2S0}	I2S transferring data @ 2 x 16 bit x 16 kHz (CONFIG.MCKFREQ = 32MDIV63, CONFIG.RATIO = 32X)		TBA		µA

5.2.1.4 PDM

Symbol	Description	Min.	Typ.	Max.	Units
I_{PDM}	PDM receiving and processing data @ 1 Msps		TBA		µA

5.2.1.5 PWM

Symbol	Description	Min.	Typ.	Max.	Units
I_{PWM0}	PWM running @ 125 kHz, fixed duty cycle		TBA		µA
I_{PWM1}	PWM running @ 16 MHz, fixed duty cycle		1160.77		µA

5.2.1.6 SAADC

Symbol	Description	Min.	Typ.	Max.	Units
I_{SAADC}	SAADC sampling @ 16 ksps, acquisition time = 20 µs		302.34		µA

5.2.1.7 TIMER

5.2.1.8 SPIM

5.2.1.9 SPIS

Symbol	Description	Min.	Typ.	Max.	Units
I_{SPIS0}	SPIS transferring data @ 2 Mbps		TBA		µA

5.2.1.10 TWIM

Symbol	Description	Min.	Typ.	Max.	Units
I_{TWIM0}	TWIM running @ 100 kbps		TBA		µA

5.2.1.11 TWIS

Symbol	Description	Min.	Typ.	Max.	Units
$I_{TWIS,RUN0}$	TWIS transferring data @ 100 kbps		TBA		µA
$I_{TWIS1,RUN1}$	TWIS transferring data @ 400 kbps,		TBA		µA

5.2.1.12 UARTE

Symbol	Description	Min.	Typ.	Max.	Units
I_{UARTE}	UARTE transferring data @ 1200 bps		TBA		µA
I_{UARTE}	UARTE transferring data @ 115200 bps		TBA		µA

5.2.1.13 WDT

Symbol	Description	Min.	Typ.	Max.	Units
I_{WDT}	WDT started		3.95		µA

5.2.1.14 Modem current consumption

Symbol	Description	B13	B20	B3	B4	Units
Modem sleep current consumption						
I_{PSM}	PSM floor current	2.7	2.7	2.7	2.7	µA
Radio resource control (RRC) mode						
I_{EDRX}	eDRX average current, 81.92 s	27	27	27	27	µA
I_{IDRX}	Idle DRX average current, 2.56 s	239	239	239	239	µA
I_{RMC_0DBM}	Uplink 180 kbit/s, Pout 0 dBm, RMC settings as per 3GPP TS 36.521-1 Annex A.2	45	45	45	45	mA
I_{RMC_10DBM}	Uplink 180 kbit/s, Pout 10 dBm, RMC settings as per 3GPP TS 36.521-1 Annex A.2	50	50	55	55	mA
I_{RMC_23DBM}	Uplink 180 kbit/s, Pout 23 dBm, RMC settings as per 3GPP TS 36.521-1 Annex A.2	105	110	140	140	mA
Modem active current consumption						
I_{TX_0DBM}	TX subframe, Pout 0 dBm	60	60	65	65	mA
I_{TX_10DBM}	TX subframe, Pout 10 dBm	80	85	95	90	mA
I_{TX_23DBM}	TX subframe, Pout 23 dBm	255	275	380	365	mA
I_{TX_-90DBM}	TX subframe, Pout -90 dBm	45	45	45	45	mA
$I_{TX\_TRANSIENT}$	TX transient	40	45	50	50	mA/µs
Modem peak current consumption						
I_{TX_PEAK}	TX subframe, Pout >21 dBm, Ant VSWR3	335	360	455	450	mA
I_{TX_PEAK}	TX subframe, Pout >20 dBm, Ant VSWR3, Vbat 3.5 V, Temp 85 °C	350	380	460	450	mA
I_{TX_PEAK}	TX subframe, Pout >20 dBm, Ant VSWR3, Vbat 3.0 V, Temp 85 °C	410	445	535	525	mA

5.3 Register description

5.3.1 POWER — Power control

The POWER module provides an interface to tasks, events, interrupt and reset related configuration settings of the power management unit.

Note: Registers INTEN on page 62, INTENSET on page 62, and INTENCLR on page 63 are the same registers (at the same address) as corresponding registers in CLOCK — Clock control on page 64.

5.3.1.1 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50005000 0x40005000	POWER	POWER : S POWER : NS	US	NA	Power control	

Table 20: Instances

Register	Offset	Security	Description
TASKS_CONSTLAT	0x78		Enable constant latency mode.
TASKS_LOWPWR	0x7C		Enable low power mode (variable latency)
SUBSCRIBE_CONSTLAT	0xF8		Subscribe configuration for task CONSTLAT
SUBSCRIBE_LOWPWR	0xFC		Subscribe configuration for task LOWPWR
EVENTS_POFWARN	0x108		Power failure warning
EVENTS_SLEEPENTER	0x114		CPU entered WFI/WFE sleep
EVENTS_SLEEPEXIT	0x118		CPU exited WFI/WFE sleep
PUBLISH_POFWARN	0x188		Publish configuration for event POFWARN
PUBLISH_SLEEPENTER	0x194		Publish configuration for event SLEEPENTER
PUBLISH_SLEEPEXIT	0x198		Publish configuration for event SLEEPEXIT
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
RESETREAS	0x400		Reset reason
POWERSTATUS	0x440		Modem domain power status
GPREGRET[0]	0x51C		General purpose retention register
GPREGRET[1]	0x520		General purpose retention register

Table 21: Register overview

5.3.1.1.1 TASKS_CONSTLAT

Address offset: 0x78

Enable constant latency mode.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_CONSTLAT			Enable constant latency mode.
			Trigger	1	Trigger task

5.3.1.1.2 TASKS_LOWPWR

Address offset: 0x7C

Enable low power mode (variable latency)

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_LOWPWR			Enable low power mode (variable latency)
			Trigger	1	Trigger task

5.3.1.1.3 SUBSCRIBE_CONSTLAT

Address offset: 0xF8

Subscribe configuration for task CONSTLAT

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task CONSTLAT will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

5.3.1.1.4 SUBSCRIBE_LOWPWR

Address offset: 0xFC

Subscribe configuration for task LOWPWR

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task LOWPWR will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

5.3.1.1.5 EVENTS_POFWARN

Address offset: 0x108

Power failure warning

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_POFWARN			Power failure warning
			NotGenerated	0	Event not generated
			Generated	1	Event generated

5.3.1.1.6 EVENTS_SLEEPENTER

Address offset: 0x114

CPU entered WFI/WFE sleep

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_SLEEPENTER			CPU entered WFI/WFE sleep
			NotGenerated	0	Event not generated
			Generated	1	Event generated

5.3.1.1.7 EVENTS_SLEEPEXIT

Address offset: 0x118

CPU exited WFI/WFE sleep

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_SLEEPEXIT			CPU exited WFI/WFE sleep
			NotGenerated	0	Event not generated
			Generated	1	Event generated

5.3.1.1.8 PUBLISH_POFWARN

Address offset: 0x188

Publish configuration for event POFWARN

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event POFWARN will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

5.3.1.1.9 PUBLISH_SLEEPENTER

Address offset: 0x194

Publish configuration for event SLEEPENTER

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event SLEEPENTER will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

5.3.1.1.10 PUBLISH_SLEEPEXIT

Address offset: 0x198

Publish configuration for event SLEEPEXIT

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event SLEEPEXIT will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

5.3.1.1.11 INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										D	C			A		
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	POFWARN			Enable or disable interrupt for event POFWARN
			Disabled	0	Disable
			Enabled	1	Enable
C	RW	SLEEPENTER			Enable or disable interrupt for event SLEEPENTER
			Disabled	0	Disable
			Enabled	1	Enable
D	RW	SLEEPEXIT			Enable or disable interrupt for event SLEEPEXIT
			Disabled	0	Disable
			Enabled	1	Enable

5.3.1.1.12 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										D	C			A		
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	POFWARN			Write '1' to enable interrupt for event POFWARN
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	SLEEPENTER			Write '1' to enable interrupt for event SLEEPENTER
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
D	RW	SLEEPEXIT			Write '1' to enable interrupt for event SLEEPEXIT
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

5.3.1.1.13 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										D	C			A		
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	POFWARN			Write '1' to disable interrupt for event POFWARN
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	SLEEPENTER			Write '1' to disable interrupt for event SLEEPENTER
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
D	RW	SLEEPEXIT			Write '1' to disable interrupt for event SLEEPEXIT
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

5.3.1.1.14 RESETREAS

Address offset: 0x400

Reset reason

Unless cleared, the RESETREAS register will be cumulative. A field is cleared by writing '1' to it. If none of the reset sources are flagged, this indicates that the chip was reset from the on-chip reset generator, which will indicate a power-on reset or a brownout reset.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID														G	F	E												D		C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	RESETPIN			Reset from pin reset detected
			NotDetected	0	Not detected
			Detected	1	Detected
B	RW	DOG			Reset from global watchdog detected
			NotDetected	0	Not detected
			Detected	1	Detected
C	RW	OFF			Reset due to wakeup from System OFF mode, when wakeup is triggered by DETECT signal from GPIO
			NotDetected	0	Not detected
			Detected	1	Detected
D	RW	DIF			Reset due to wakeup from System OFF mode, when wakeup is triggered by entering debug interface mode
			NotDetected	0	Not detected
			Detected	1	Detected
E	RW	SREQ			Reset from AIRCR.SYSRESETREQ detected
			NotDetected	0	Not detected
			Detected	1	Detected
F	RW	LOCKUP			Reset from CPU lock-up detected

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID														G	F	E												D		C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
			NotDetected	0	Not detected
			Detected	1	Detected
G	RW	CTRLAP			Reset triggered through CTRL-AP
			NotDetected	0	Not detected
			Detected	1	Detected

5.3.1.1.15 POWERSTATUS

Address offset: 0x440

Modem domain power status

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	LTEMODEM			LTE modem domain status
			OFF	0	LTE modem domain is powered off
			ON	1	LTE modem domain is powered on

5.3.1.1.16 GPREGRET[n] (n=0..1)

Address offset: 0x51C + (n × 0x4)

General purpose retention register

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	GPREGRET			General purpose retention register This register is a retained register

5.3.2 CLOCK — Clock control

The CLOCK module provides one of the interfaces to power and clock management configuration settings.

Through CLOCK module it is able to configure the following:

- LFCLK clock source setup
- LFCLK and HFCLK status
- Tasks and events
- Interrupts
- Reset

Note: Registers INTEN on page 68, INTENSET on page 69, and INTENCLR on page 69 are the same registers (at the same address) as corresponding registers in POWER — Power control on page 58.

5.3.2.1 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50005000 0x40005000	CLOCK	CLOCK : S CLOCK : NS	US	NA	Clock control	

Table 22: Instances

Register	Offset	Security	Description
TASKS_HFCLKSTART	0x000		Start HFCLK source
TASKS_HFCLKSTOP	0x004		Stop HFCLK source
TASKS_LFCLKSTART	0x008		Start LFCLK source
TASKS_LFCLKSTOP	0x00C		Stop LFCLK source
SUBSCRIBE_HFCLKSTART	0x080		Subscribe configuration for task HFCLKSTART
SUBSCRIBE_HFCLKSTOP	0x084		Subscribe configuration for task HFCLKSTOP
SUBSCRIBE_LFCLKSTART	0x088		Subscribe configuration for task LFCLKSTART
SUBSCRIBE_LFCLKSTOP	0x08C		Subscribe configuration for task LFCLKSTOP
EVENTS_HFCLKSTARTED	0x100		HFCLK oscillator started
EVENTS_LFCLKSTARTED	0x104		LFCLK started
PUBLISH_HFCLKSTARTED	0x180		Publish configuration for event HFCLKSTARTED
PUBLISH_LFCLKSTARTED	0x184		Publish configuration for event LFCLKSTARTED
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
INTPEND	0x30C		Pending interrupts
HFCLKRUN	0x408		Status indicating that HFCLKSTART task has been triggered
HFCLKSTAT	0x40C		The register shows if HFXO has been requested by triggering HFCLKSTART task and if it has been started (STATE)
LFCLKRUN	0x414		Status indicating that LFCLKSTART task has been triggered
LFCLKSTAT	0x418		The register shows which LFCLK source has been requested (SRC) when triggering LFCLKSTART task and if the source has been started (STATE)
LFCLKSRCCOPY	0x41C		Copy of LFCLKSRC register, set after LFCLKSTART task has been triggered
LFCLKSRC	0x518		Clock source for the LFCLK. LFCLKSTART task starts starts a clock source selected with this register.

Table 23: Register overview

5.3.2.1.1 TASKS_HFCLKSTART

Address offset: 0x000

Start HFCLK source

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_HFCLKSTART			Start HFCLK source
			Trigger	1	Trigger task

5.3.2.1.2 TASKS_HFCLKSTOP

Address offset: 0x004

Stop HFCLK source

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_HFCLKSTOP			Stop HFCLK source
			Trigger	1	Trigger task

5.3.2.1.3 TASKS_LFCLKSTART

Address offset: 0x008

Start LFCLK source

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_LFCLKSTART			Start LFCLK source
			Trigger	1	Trigger task

5.3.2.1.4 TASKS_LFCLKSTOP

Address offset: 0x00C

Stop LFCLK source

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_LFCLKSTOP			Stop LFCLK source
			Trigger	1	Trigger task

5.3.2.1.5 SUBSCRIBE_HFCLKSTART

Address offset: 0x080

Subscribe configuration for task HFCLKSTART

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task HFCLKSTART will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

5.3.2.1.6 SUBSCRIBE_HFCLKSTOP

Address offset: 0x084

Subscribe configuration for task HFCLKSTOP

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task HFCLKSTOP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

5.3.2.1.7 SUBSCRIBE_LFCLKSTART

Address offset: 0x088

Subscribe configuration for task LFCLKSTART

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task LFCLKSTART will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

5.3.2.1.8 SUBSCRIBE_LFCLKSTOP

Address offset: 0x08C

Subscribe configuration for task LFCLKSTOP

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task LFCLKSTOP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

5.3.2.1.9 EVENTS_HFCLKSTARTED

Address offset: 0x100

HFCLK oscillator started

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_HFCLKSTARTED			HFCLK oscillator started
			NotGenerated	0	Event not generated
			Generated	1	Event generated

5.3.2.1.10 EVENTS_LFCLKSTARTED

Address offset: 0x104

LFCLK started

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_LFCLKSTARTED			LFCLK started
			NotGenerated	0	Event not generated
			Generated	1	Event generated

5.3.2.1.11 PUBLISH_HFCLKSTARTED

Address offset: 0x180

Publish configuration for event HFCLKSTARTED

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event HFCLKSTARTED will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

5.3.2.1.12 PUBLISH_LFCLKSTARTED

Address offset: 0x184

Publish configuration for event LFCLKSTARTED

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event LFCLKSTARTED will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

5.3.2.1.13 INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	HFCLKSTARTED			Enable or disable interrupt for event HFCLKSTARTED
			Disabled	0	Disable
			Enabled	1	Enable
B	RW	LFCLKSTARTED			Enable or disable interrupt for event LFCLKSTARTED
			Disabled	0	Disable
			Enabled	1	Enable

5.3.2.1.14 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	HFCLKSTARTED			Write '1' to enable interrupt for event HFCLKSTARTED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	LFCLKSTARTED			Write '1' to enable interrupt for event LFCLKSTARTED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

5.3.2.1.15 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	HFCLKSTARTED			Write '1' to disable interrupt for event HFCLKSTARTED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	LFCLKSTARTED			Write '1' to disable interrupt for event LFCLKSTARTED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

5.3.2.1.16 INTPEND

Address offset: 0x30C

Pending interrupts

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	HFCLKSTARTED			Read pending status of interrupt for event HFCLKSTARTED
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
B	R	LFCLKSTARTED			Read pending status of interrupt for event LFCLKSTARTED
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending

5.3.2.1.17 HFCLKRUN

Address offset: 0x408

Status indicating that HFCLKSTART task has been triggered

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	STATUS			HFCLKSTART task triggered or not
			NotTriggered	0	Task not triggered
			Triggered	1	Task triggered

5.3.2.1.18 HFCLKSTAT

Address offset: 0x40C

The register shows if HFXO has been requested by triggering HFCLKSTART task and if it has been started (STATE)

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																B																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	SRC			Active clock source
			HFXO	1	HFXO - 64 MHz clock derived from external 32 MHz crystal oscillator
B	R	STATE			HFCLK state
			NotRunning	0	HFXO has not been started or HFCLKSTOP task has been triggered
			Running	1	HFXO has been started (HFCLKSTARTED event has been generated)

5.3.2.1.19 LFCLKRUN

Address offset: 0x414

Status indicating that LFCLKSTART task has been triggered

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	STATUS			LFCLKSTART task triggered or not
			NotTriggered	0	Task not triggered
			Triggered	1	Task triggered

5.3.2.1.20 LFCLKSTAT

Address offset: 0x418

The register shows which LFCLK source has been requested (SRC) when triggering LFCLKSTART task and if the source has been started (STATE)

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																B															A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	SRC			Active clock source
			RFU	0	Reserved for future use
			LFRC	1	32.768 kHz RC oscillator
			LFXO	2	32.768 kHz crystal oscillator
B	R	STATE			LFCLK state
			NotRunning	0	Requested LFCLK source has not been started or LFCLKSTOP task has been triggered
			Running	1	Requested LFCLK source has been started (LFCLKSTARTED event has been generated)

5.3.2.1.21 LFCLKSRCCOPY

Address offset: 0x41C

Copy of LFCLKSRC register, set after LFCLKSTART task has been triggered

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															A	A
Reset 0x00000001	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

ID	Acce	Field	Value ID	Value	Description
A	R	SRC			Clock source
			RFU	0	Reserved for future use
			LFRC	1	32.768 kHz RC oscillator
			LFXO	2	32.768 kHz crystal oscillator

5.3.2.1.22 LFCLKSRC

Address offset: 0x518

Clock source for the LFCLK. LFCLKSTART task starts starts a clock source selected with this register.

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				A A
Reset 0x00000001				**0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1**
ID	Acce Field	Value ID	Value	Description
A	RW SRC			Clock source
		RFU	0	Reserved for future use (equals selecting LFRC)
		LFRC	1	32.768 kHz RC oscillator
		LFXO	2	32.768 kHz crystal oscillator

5.3.3 REGULATORS — Voltage regulators control

The REGULATORS module provides an interface to certain configuration settings of on-chip voltage regulators.

5.3.3.1 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50004000 0x40004000	REGULATORS	REGULATORS : S REGULATORS : NS	US	NA	Regulator configuration	

Table 24: Instances

Register	Offset	Security	Description
SYSTEMOFF	0x500		System OFF register
DCDCEN	0x578		Enable DC/DC mode of the main voltage regulator

Table 25: Register overview

5.3.3.1.1 SYSTEMOFF

Address offset: 0x500

System OFF register

Bit number				31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
ID				A
Reset 0x00000000				**0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0**
ID	Acce Field	Value ID	Value	Description
A	W SYSTEMOFF			Enable System OFF mode
		Enable	1	Enable System OFF mode

5.3.3.1.2 DCDCEN

Address offset: 0x578

Enable DC/DC mode of the main voltage regulator

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	DCDCEN			Enable DC/DC converter
			Disabled	0	DC/DC mode is disabled
			Enabled	1	DC/DC mode is enabled

6 Peripherals

6.1 CRYPTOCELL — ARM TrustZone CryptoCell 310

ARM® TrustZone® CryptoCell 310 (CRYPTOCELL) is a security subsystem which provides root of trust (RoT) and cryptographic services for a device.

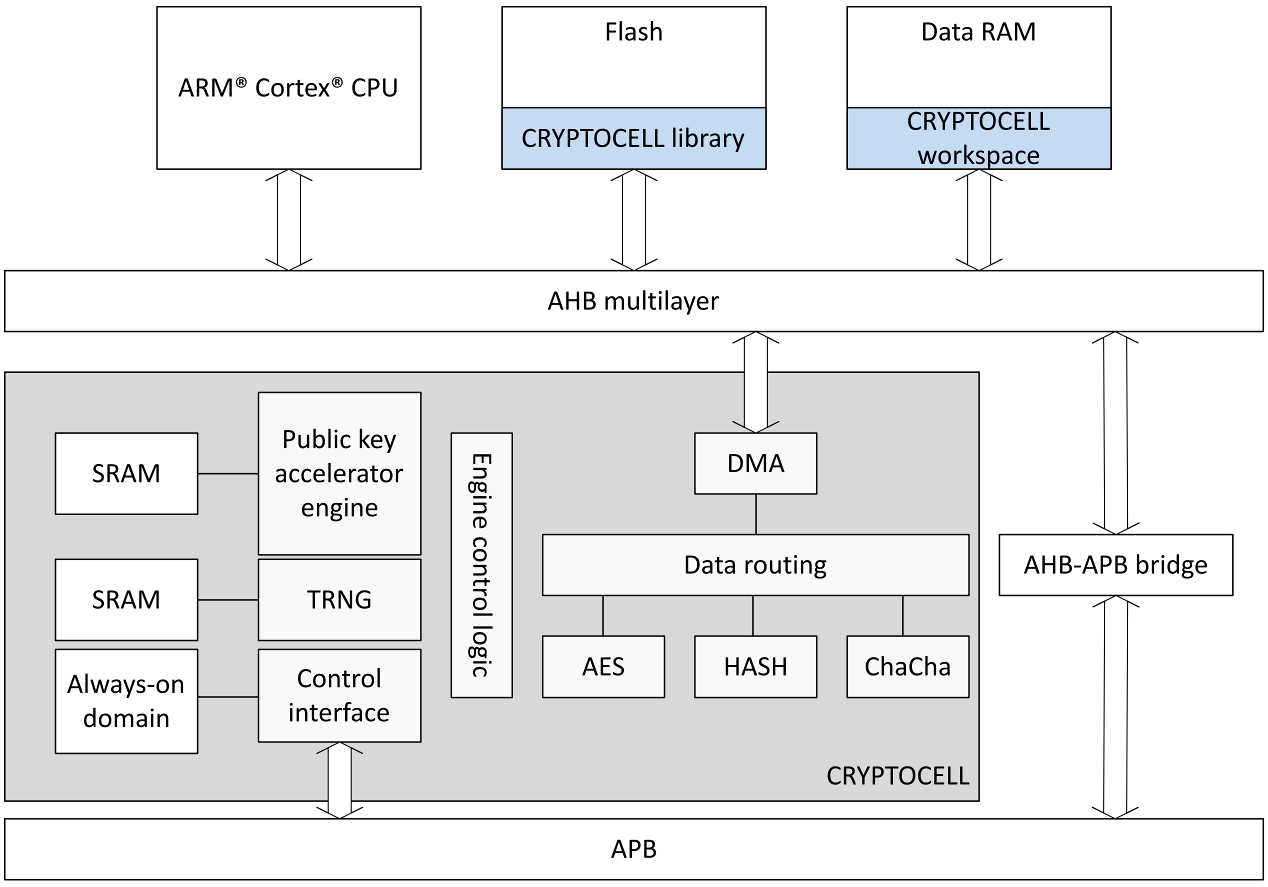


Figure 11: Block diagram for CRYPTOCELL

The following cryptographic features are provided:

- True random number generator (TRNG) compliant with NIST 800-90B[7], AIS-31, and FIPS 140-2/3[7].
- Pseudorandom number generator (PRNG) using underlying AES engine compliant with NIST 800-90A
- RSA public key cryptography
  - Up to 2048-bit key size
  - PKCS#1 v2.1/v1.5
  - Optional CRT support
- Elliptic curve cryptography (ECC)
  - NIST FIPS 186-4 recommended curves using pseudorandom parameters, up to 521 bits:
    - Prime field: P-192, P-224, P-256, P-384, P-521
  - SEC 2 recommended curves using pseudorandom parameters, up to 521 bits:

[7] Not finalized at time of publishing (draft)

        - Prime field: secp160r1, secp192r1, secp224r1, secp256r1, secp384r1, secp521r1
    - Koblitz curves using fixed parameters, up to 256 bits:
        - Prime field: secp160k1, secp192k1, secp224k1, secp256k1
    - Edwards/Montgomery curves:
        - Ed25519, Curve25519
    - ECDH/ECDSA support
- Secure remote password protocol (SRP)
    - Up to 3072-bit operations
- Hashing functions
    - SHA-1, SHA-2 up to 256 bits
    - Keyed-hash message authentication code (HMAC)
- AES symmetric encryption
    - General purpose AES engine (encrypt/decrypt, sign/verify)
    - 128-bit key size
    - Supported encryption modes: ECB, CBC, CMAC/CBC-MAC, CTR, CCM/CCM*
- ChaCha20/Poly1305 symmetric encryption
    - Supported key size: 128 and 256 bits
    - Authenticated encryption with associated data (AEAD) mode

6.1.1 Usage

The CRYPTOCELL state is controlled via a register interface. The cryptographic functions of CRYPTOCELL are accessible by using a software library provided in the device SDK, not directly via a register interface.

To enable CRYPTOCELL, use register ENABLE on page 77.

6.1.2 Always-on (AO) power domain

The CRYPTOCELL subsystem has an internal always-on (AO) power domain for retaining device secrets when CRYPTOCELL is disabled.

The following information is retained by the AO power domain:

- 4 bits indicating the configured CRYPTOCELL life-cycle state (LCS)
- 1 bit indicating if RTL key K_{PRTL} is available for use
- 128-bit device root key K_{DR}

A reset from any reset source will erase the content in the AO power domain.

6.1.3 Lifecycle state (LCS)

Lifecycle refers to multiple states a device goes through during its lifetime. Two valid lifecycle states are offered for the device - debug and secure.

The CRYPTOCELL subsystem lifecycle state (LCS) is controlled through register . A valid LCS is configured by writing either value `Debug` or `Secure` into the LCS field of this register. A correctly configured LCS can be validated by reading back the read-only field LCS_IS_VALID from the abovementioned register. The LCS_IS_VALID field value will change from `Invalid` to `Valid` once a valid LCS value has been written.

LCS field value	LCS_IS_VALID field value	Description
Secure	Invalid	Default reset value indicating that LCS has not been configured.
Secure	Valid	LCS set to secure mode, and LCS is valid. Registers HOST_IOT_KDR[0..3] can only be written once per reset cycle. Any additional writes will be ignored.
Debug	Valid	LCS set to debug mode, and LCS is valid. Registers HOST_IOT_KDR[0..3] can be written multiple times.

Table 26: Lifecycle states

6.1.4 Cryptographic key selection

The CRYPTOCELL subsystem can be instructed to operate on different cryptographic keys.

Through register , the following key types can be selected for cryptographic operations:

- RTL key K_{PRTL}
- Device root key K_{DR}
- Session key

K_{PRTL} and K_{DR} are configured as part of the CRYPTOCELL initialization process, while session keys are provided by the application through the software library API.

6.1.4.1 RTL key

The ARM® TrustZone® CryptoCell 310 IP contains one hard-coded RTL key referred to as K_{PRTL}. This key is set to the same value for all devices with the same part code in the hardware design and cannot be changed.

The K_{PRTL} key can be requested for use in cryptographic operations by the CRYPTOCELL, without revealing the key value itself. Access to use of K_{PRTL} in cryptographic operations can be disabled until next reset by writing to register . If a locked K_{PRTL} key is requested for use, a zero vector key will be routed to the AES engine instead.

6.1.4.2 Device root key

The device root key K_{DR} is a 128-bit AES key programmed into the CRYPTOCELL subsystem using firmware. It is retained in the AO power domain until the next reset.

Once configured, it is possible to perform cryptographic operations using the the CRYPTOCELL subsystem where K_{DR} is selected as key input without having access to the key value itself. The K_{DR} key value must be written to registers HOST_IOT_KDR[0..3]. These 4 registers are write-only if LCS is set to debug mode, and write-once if LCS is set to secure mode. The K_{DR} key value is successfully retained when the read-back value of register changes to `1`.

6.1.5 Direct memory access (DMA)

The CRYPTOCELL subsystem implements direct memory access (DMA) for accessing memory without CPU intervention.

The following table shows which memory type(s) can be accessed using the DMA:

Any data stored in memory type(s) not accessible by the DMA engine must be copied to SRAM before it can be processed by the CRYPTOCELL subsystem. Maximum DMA transaction size is limited to 2^{16}-1 bytes.

6.1.6 Standards

ARM® TrustZone® CryptoCell 310 (CRYPTOCELL) supports a number of cryptography standards.

Algorithm family	Identification code	Document title
TRNG	NIST SP 800-90B	*Recommendation for the Entropy Sources Used for Random Bit Generation*
	AIS-31	*A proposal for: Functionality classes and evaluation methodology for physical random number generators*
	FIPS 140-2	*Security Requirements for Cryptographic Modules*
PRNG	NIST SP 800-90A	*Recommendation for Random Number Generation Using Deterministic Random Bit Generators*
Stream cipher	Chacha	*ChaCha, a variant of Salsa20*, Daniel J. Bernstein, January 28th 2008
MAC	Poly1305	*The Poly1305-AES message-authentication code*, Daniel J. Bernstein *Cryptography in NaCl*, Daniel J. Bernstein
Key agreement	SRP	*The Secure Remote Password Protocol*, Thomas Wu, November 11th 1997
AES	FIPS-197	*Advanced Encryption Standard (AES)*
	NIST SP 800-38A	*Recommendation for Block Cipher Modes of Operation - Methods and Techniques*
	NIST SP 800-38B	*Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*
	NIST SP 800-38C	*Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*
	ISO/IEC 9797-1	AES CBC-MAC per ISO/IEC 9797-1 MAC algorithm 1
	IEEE 802.15.4-2011	*IEEE Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, Annex B.4: *Specification of generic CCM* mode of operation*
Hash	FIPS 180-3	Secure Hash Standard (SHA1, SHA-224, SHA-256)
	RFC2104	*HMAC: Keyed-Hashing for Message Authentication*
RSA	PKCS#1	*Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications* v1.5/2.1
Diffie-Hellman	ANSI X9.42	*Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography*
	PKCS#3	*Diffie-Hellman Key-Agreement Standard*
ECC	ANSI X9.63	*Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography*
	IEEE 1363	*Standard Specifications for Public-Key Cryptography*
	ANSI X9.62	*Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*
	Ed25519	Edwards-curve, *Ed25519: high-speed high-security signatures*, Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang
	Curve25519	Montgomery curve, *Curve25519: new Diffie-Hellman speed records*, Daniel J. Bernstein
	FIPS 186-4	*Digital Signature Standard (DSS)*
	SEC 2	*Recommended Elliptic Curve Domain Parameters*, Certicom Research
	NIST SP 800-56A rev. 2	*Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*
General	FIPS 140-2	*Security Requirements for Cryptographic Modules*

Table 27: CRYPTOCELL cryptography standards

6.1.7 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50840000	CRYPTOCELL	CRYPTOCELL	S	NSA	CryptoCell sub-system control interface	

Table 28: Instances

Register	Offset	Security	Description
ENABLE	0x500		Enable CRYPTOCELL subsystem

Table 29: Register overview

6.1.7.1 ENABLE

Address offset: 0x500

Enable CRYPTOCELL subsystem

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	ENABLE			Enable or disable the CRYPTOCELL subsystem
			Disabled	0	CRYPTOCELL subsystem disabled
			Enabled	1	CRYPTOCELL subsystem enabled
					When enabled the CRYPTOCELL subsystem can be initialized and controlled through the CryptoCell firmware API

6.1.8 Host interface

This chapter describe host registers used for controlling the CRYPTOCELL subsystem behavior.

6.1.8.1 HOST_RGF block

The HOST_RGF block contains registers for configuring LCS and device root key K_{DR}, in addition to selecting which cryptographic key is connected to the AES engine.

6.2 DPPI - Distributed programmable peripheral interconnect

The distributed programmable peripheral interconnect (DPPI) enables peripherals to interact autonomously with each other, using tasks and events, without any intervention from the CPU. DPPI allows precise synchronization between peripherals when real-time application constraints exist, and eliminates the need of CPU involvement to implement behavior which can be predefined using the DPPI.

DPPI has the following features:

- Peripheral tasks can subscribe to channels
- Peripheral events can be published on channels
- Publish/subscribe pattern enabling multiple connection options:
    - One-to-one
    - One-to-many
    - Many-to-one
    - Many-to-many

The DPPI system consists of several PPIBus modules, which are connected to a fixed number of DPPI channels and a DPPI controller (DPPIC):

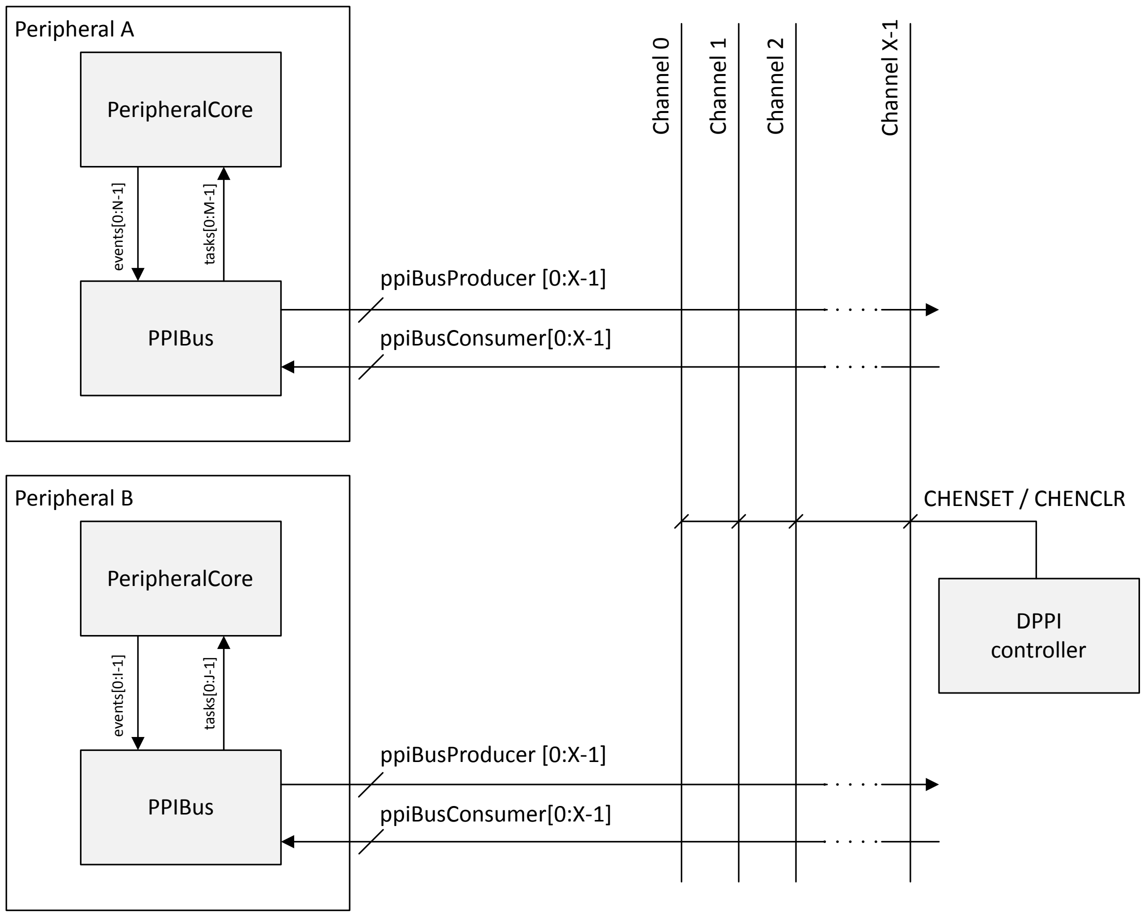


Figure 12: DPPI overview

6.2.1 Subscribing to and publishing on channels

The PPIBus can route peripheral events onto the channels (publishing), or route events from the channels into peripheral tasks (subscribing).

All peripherals include:

- One subscribe register per task
- One publish register per event

Publish and subscribe registers use channel index field to determine the channel to which the event is published or tasks subscribed. In addition, there is an enable bit for the subscribe and publish registers that needs to be enabled before the subscription or publishing takes effect.

One event can trigger multiple tasks by subscribing different tasks to the same channel. Similarly, one task could be triggered by multiple events by publishing different events to the same channel. For advanced use cases, multiple events and multiple tasks could be connected to the same channel forming a many-to-many connection. If multiple events are published on the same channel at the same time, the events will be merged and only one event is routed through the DPPI system.

DPPI events flow on page 80 shows how peripheral events are routed onto different channels based on the publish registers.

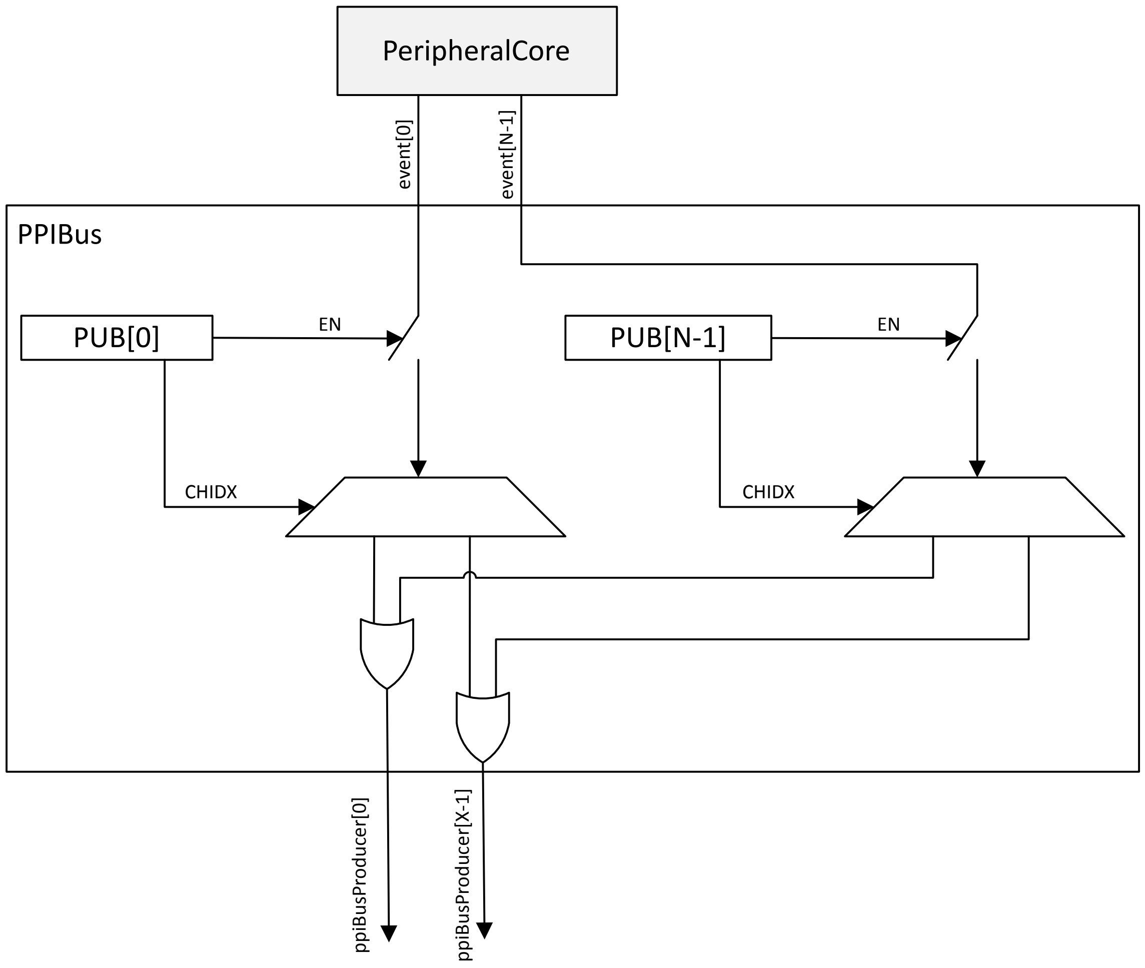


Figure 13: DPPI events flow

DPPI tasks flow on page 81 shows how peripheral tasks are triggered from different channels based on the subscribe registers.

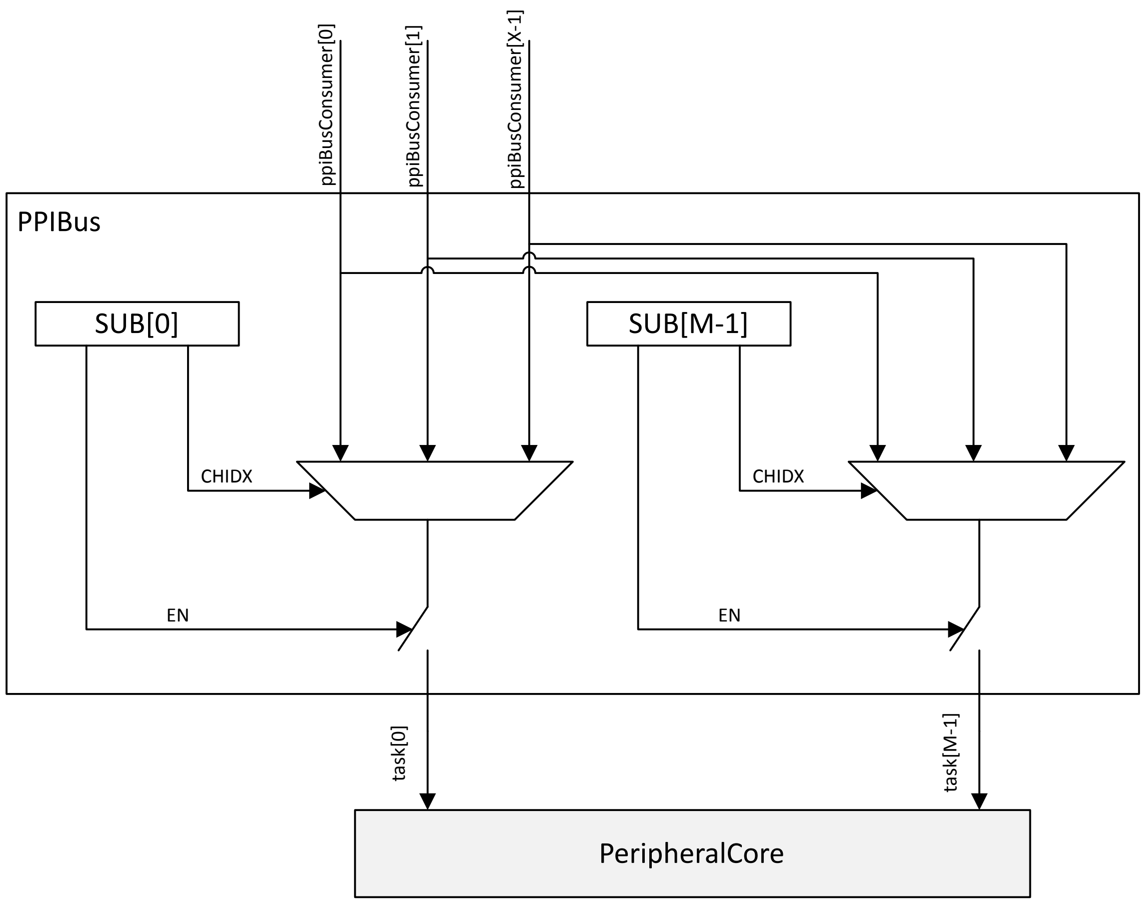


Figure 14: DPPI tasks flow

6.2.2 DPPI controller

Enabling/disabling and control of DPPI channels are handled locally at every peripheral and through the centralized DPPI controller peripheral.

There are two ways of enabling and disabling global channels using the DPPIC:

- Enable or disable channels individually using the CHEN, CHENSET and CHENCLR registers.
- Enable or disable channels in channel groups using the groups' ENABLE and DISABLE tasks. Prior triggering these tasks, it needs to be defined which channels belong to which channel groups.

Important: When a channel belongs to two (or more) groups, for example group m and n, and the tasks CHG[m].EN and CHG[n].DIS occur simultaneously (m and n can be equal or different), the CHG[m].EN task on that channel has priority. Meaning that enable tasks are prioritized over disable tasks.

DPPIC tasks (for example CHG[0].EN) can be triggered through the DPPI system like any other task, which means they can be hooked to a DPPI channel through the subscribe registers.

In order to write to CHG[x], the corresponding CHG[x].EN and CHG[x].DIS subscribe registers must be disabled. Writes to CHG[x] are ignored when either subscribe register is enabled.

6.2.3 Connection examples

DPPI offers several connection options. Examples are given for how to create one-to-one and many-to-many connections.

One-to-one connection

This example shows how to create a one-to-one connection between TIMER compare register and SAADC start task.

The channel configuration is set up first, TIMER0 will publish its COMPARE0 event on channel 0, and SAADC will subscribe its START task to events on the same channel. After that, the channel is enabled in the DPPI controller.

```
NRF_TIMER0->PUBLISH_COMPARE0 = (DPPI_PUB_CHIDX_Ch0) |
                               (DPPI_PUB_EN_Msk);
NRF_SAADC->SUBSCRIBE_START   = (DPPI_SUB_CHIDX_Ch0) |
                               (DPPI_SUB_EN_Msk);

NRF_DPPIC->CHENSET = (DPPI_CHENSET_CH0_Set << DPPI_CHENSET_CH0_Pos);
```

Many-to-many connection

The example shows how to create a many-to-many connection, also showcasing the use of the channel group functionality of the DPPI controller.

A channel group, including only channel 0, is set up first. Then GPIOTE and TIMER0 configure their IN0 and COMPARE0 events respectively to be published on channel 0, while SAADC configures its START task to subscribe to events on channel 0. The DPPI controller configures its CHG0 disable task to subscribe to events on channel 0. This will effectively disable channel 0 after an event is received on channel 0. Finally, channel 0 is enabled using the DPPI controller task to enable a channel group.

```
NRF_DPPIC->CHG[0] = (DPPI_CHG_CH0_Included << PPI_CHG_CH0_Pos);

NRF_GPIOTE->PUBLISH_IN0          = (DPPI_PUB_CHIDX_Ch0) |
                                   (DPPI_PUB_EN_Msk);
NRF_TIMER0->PUBLISH_COMPARE0     = (DPPI_PUB_CHIDX_Ch0) |
                                   (DPPI_PUB_EN_Msk);
NRF_SAADC->SUBSCRIBE_START       = (DPPI_SUB_CHIDX_Ch0) |
                                   (DPPI_SUB_EN_Msk);
NRF_DPPIC->SUBSCRIBE_CHG[0].DIS = (DPPI_SUB_CHIDX_Ch0) |
                                   (DPPI_SUB_EN_Msk);

NRF_DPPIC->TASK_CHG[0].EN = 1;
```

6.2.4 Special considerations for system implementing TrustZone for Cortex-M® processors

In a system implementing the TrustZone for Cortex-M® technology, DPPI channels can be defined as "Secure" or "Non-Secure" using the SPU - System protection unit on page 257 :

- A peripheral configured with a non-secure security attribute will only be able to subscribe or publish to non secure DPPI channels.
- A peripheral configured as secure will be able to access all DPPI channels

The DPPIC is implemented as a "split-security" peripheral. It is therefore accessible by both secure and non-secure accesses but the DPPIC behaves differently depending of the access type :

- A non-secure peripheral access will only be able to configure and control DPPI channels defined as non-secure in the SPU.DPPI.PERM[] register(s).
- A secure peripheral access can control all the DPPI channels, independently of the SPU.DPPI.PERM[] register(s).

The DPPIC allows the creation of a group of channels to be able to simultaneously enable or disable all channels within a group . The security attribute of a group of channels (secure or non-secure) is defined as follows:

- If all the channels (enabled or not) of a group are non-secure, then the group is considered as non-secure
- If at least one of the channels (enabled or not) of the group is secure, then the group is considered as secure

A non-secure access to a DPPIC register or a bitfield controlling a channel marked as secure in SPU.DPPI[].PERM register(s) will be ignored :

- Write accesses will have no effect
- Read accesses will always return a zero value

No exception is triggered when a non-secure access targets a register or bitfield controlling a secure channel.

For example, if the bit *i* is set in the SPU.DPPI[0].PERM register (declaring DPPI channel *i* as secure), then

- Non-secure write accesses to CHEN, CHENSET and CHENCLR registers will not be able to write to bit *i* of those registers
- Non-secure write accesses to TASK_CHG[*j*].EN and TASK_CHG[*j*].DIS registers will be ignored if the channel group *j* contains at least a channel defined as secure (it can be the channel *i* itself or any channel declared as secure)
- Non-secure read accesses to registers CHEN, CHENSET and CHENCLR will always read a 0 for the bit at position *i*

For the channel configuration registers (DPPIC.CHG[]), access from non-secure code is only possible if the included channels are all non-secure, whether the channels are enabled or not. If a DPPIC.CHG[*g*] register included one or more secure channel, then the group *g* is considered as secure and only a secure transfer can read or write DPPIC.CHG[*g*]. A non-secure write access will be ignored and a non-secure read access will return 0.

The DPPIC can subscribe to both secure or non-secure channel through the SUBSCRIBE_CHG[] registers in order to trigger task for enabling or disabling groups of channels. But an event from a non-secure channel will be ignored if the group subscribing to this channel is secure. A event from a secure channel can trigger both secure and non-secure tasks.

6.2.5 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50017000 0x40017000	DPPIC	DPPIC : S DPPIC : NS	SPLIT	NA	DPPI controller	

Table 30: Instances

Register	Offset	Security	Description
TASKS_CHG[0].EN	0x000		Enable channel group 0
TASKS_CHG[0].DIS	0x004		Disable channel group 0

Register	Offset	Security	Description
TASKS_CHG[1].EN	0x008		Enable channel group 1
TASKS_CHG[1].DIS	0x00C		Disable channel group 1
TASKS_CHG[2].EN	0x010		Enable channel group 2
TASKS_CHG[2].DIS	0x014		Disable channel group 2
TASKS_CHG[3].EN	0x018		Enable channel group 3
TASKS_CHG[3].DIS	0x01C		Disable channel group 3
TASKS_CHG[4].EN	0x020		Enable channel group 4
TASKS_CHG[4].DIS	0x024		Disable channel group 4
TASKS_CHG[5].EN	0x028		Enable channel group 5
TASKS_CHG[5].DIS	0x02C		Disable channel group 5
SUBSCRIBE_CHG[0].EN	0x080		Subscribe configuration for task CHG[0].EN
SUBSCRIBE_CHG[0].DIS	0x084		Subscribe configuration for task CHG[0].DIS
SUBSCRIBE_CHG[1].EN	0x088		Subscribe configuration for task CHG[1].EN
SUBSCRIBE_CHG[1].DIS	0x08C		Subscribe configuration for task CHG[1].DIS
SUBSCRIBE_CHG[2].EN	0x090		Subscribe configuration for task CHG[2].EN
SUBSCRIBE_CHG[2].DIS	0x094		Subscribe configuration for task CHG[2].DIS
SUBSCRIBE_CHG[3].EN	0x098		Subscribe configuration for task CHG[3].EN
SUBSCRIBE_CHG[3].DIS	0x09C		Subscribe configuration for task CHG[3].DIS
SUBSCRIBE_CHG[4].EN	0x0A0		Subscribe configuration for task CHG[4].EN
SUBSCRIBE_CHG[4].DIS	0x0A4		Subscribe configuration for task CHG[4].DIS
SUBSCRIBE_CHG[5].EN	0x0A8		Subscribe configuration for task CHG[5].EN
SUBSCRIBE_CHG[5].DIS	0x0AC		Subscribe configuration for task CHG[5].DIS
CHEN	0x500		Channel enable register
CHENSET	0x504		Channel enable set register
CHENCLR	0x508		Channel enable clear register
CHG[0]	0x800		Channel group 0 Note: Writes to this register is ignored if either SUBSCRIBE_CHG[0].EN/DIS are enabled.
CHG[1]	0x804		Channel group 1 Note: Writes to this register is ignored if either SUBSCRIBE_CHG[1].EN/DIS are enabled.
CHG[2]	0x808		Channel group 2 Note: Writes to this register is ignored if either SUBSCRIBE_CHG[2].EN/DIS are enabled.
CHG[3]	0x80C		Channel group 3 Note: Writes to this register is ignored if either SUBSCRIBE_CHG[3].EN/DIS are enabled.
CHG[4]	0x810		Channel group 4 Note: Writes to this register is ignored if either SUBSCRIBE_CHG[4].EN/DIS are enabled.
CHG[5]	0x814		Channel group 5 Note: Writes to this register is ignored if either SUBSCRIBE_CHG[5].EN/DIS are enabled.

Table 31: Register overview

6.2.5.1 TASKS_CHG[n].EN (n=0..5)

Address offset: 0x000 + (n × 0x8)

Enable channel group n

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	EN			Enable channel group n
			Trigger	1	Trigger task

6.2.5.2 TASKS_CHG[n].DIS (n=0..5)

Address offset: 0x004 + (n × 0x8)

Disable channel group n

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	DIS			Disable channel group n
			Trigger	1	Trigger task

6.2.5.3 SUBSCRIBE_CHG[n].EN (n=0..5)

Address offset: 0x080 + (n × 0x8)

Subscribe configuration for task CHG[n].EN

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task CHG[n].EN will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.2.5.4 SUBSCRIBE_CHG[n].DIS (n=0..5)

Address offset: 0x084 + (n × 0x8)

Subscribe configuration for task CHG[n].DIS

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task CHG[n].DIS will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.2.5.5 CHEN

Address offset: 0x500

Channel enable register

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-P	RW	CH[i] (i=0..15)			Enable or disable channel i
			Disabled	0	Disable channel
			Enabled	1	Enable channel

6.2.5.6 CHENSET

Address offset: 0x504

Channel enable set register

Read: reads value of CH{i} field in CHEN register.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-P	RW	CH[i] (i=0..15)			Channel i enable set register. Writing '0' has no effect
			Disabled	0	Read: channel disabled
			Enabled	1	Read: channel enabled
			Set	1	Write: Enable channel

6.2.5.7 CHENCLR

Address offset: 0x508

Channel enable clear register

Read: reads value of CH{i} field in CHEN register.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-P	RW	CH[i] (i=0..15)			Channel i enable clear register. Writing '0' has no effect
			Disabled	0	Read: channel disabled
			Enabled	1	Read: channel enabled
			Clear	1	Write: disable channel

6.2.5.8 CHG[n] (n=0..5)

Address offset: 0x800 + (n × 0x4)

Channel group n

Note: Writes to this register is ignored if either SUBSCRIBE_CHG[n].EN/DIS are enabled.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-P	RW	CH[i] (i=0..15)			Include or exclude channel i
			Excluded	0	Exclude
			Included	1	Include

6.3 EGU — Event generator unit

The Event generator unit (EGU) provides support for inter-layer signaling. This means support for atomic triggering of both CPU execution and hardware tasks from both firmware (by CPU) and hardware (by PPI). This feature can, for instance, be used for triggering CPU execution at a lower priority execution from a higher priority execution, or to handle a peripheral's ISR execution at a lower priority for some of its events. However, triggering any priority from any priority is possible.

Listed here are the main EGU features:

- Enables SW triggering of interrupts
- Separate interrupt vectors for every EGU instance
- Up to 16 separate event flags per interrupt for multiplexing

Each instance of The EGU implements a set of tasks which can individually be triggered to generate the corresponding event, i.e., the corresponding event for TASKS_TRIGGER[n] is EVENTS_TRIGGERED[n].

Refer to Instances on page 87 for a list of the various EGU instances

6.3.1 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x5001B000 0x4001B000	EGU	EGU0 : S EGU0 : NS	US	NA	Event generator unit 0	
0x5001C000 0x4001C000	EGU	EGU1 : S EGU1 : NS	US	NA	Event generator unit 1	
0x5001D000 0x4001D000	EGU	EGU2 : S EGU2 : NS	US	NA	Event generator unit 2	
0x5001E000 0x4001E000	EGU	EGU3 : S EGU3 : NS	US	NA	Event generator unit 3	
0x5001F000 0x4001F000	EGU	EGU4 : S EGU4 : NS	US	NA	Event generator unit 4	
0x50020000 0x40020000	EGU	EGU5 : S EGU5 : NS	US	NA	Event generator unit 5	

Table 32: Instances

Register	Offset	Security	Description
TASKS_TRIGGER[0]	0x000		Trigger 0 for triggering the corresponding TRIGGERED[0] event
TASKS_TRIGGER[1]	0x004		Trigger 1 for triggering the corresponding TRIGGERED[1] event
TASKS_TRIGGER[2]	0x008		Trigger 2 for triggering the corresponding TRIGGERED[2] event
TASKS_TRIGGER[3]	0x00C		Trigger 3 for triggering the corresponding TRIGGERED[3] event
TASKS_TRIGGER[4]	0x010		Trigger 4 for triggering the corresponding TRIGGERED[4] event
TASKS_TRIGGER[5]	0x014		Trigger 5 for triggering the corresponding TRIGGERED[5] event
TASKS_TRIGGER[6]	0x018		Trigger 6 for triggering the corresponding TRIGGERED[6] event
TASKS_TRIGGER[7]	0x01C		Trigger 7 for triggering the corresponding TRIGGERED[7] event

Register	Offset	Security	Description
TASKS_TRIGGER[8]	0x020		Trigger 8 for triggering the corresponding TRIGGERED[8] event
TASKS_TRIGGER[9]	0x024		Trigger 9 for triggering the corresponding TRIGGERED[9] event
TASKS_TRIGGER[10]	0x028		Trigger 10 for triggering the corresponding TRIGGERED[10] event
TASKS_TRIGGER[11]	0x02C		Trigger 11 for triggering the corresponding TRIGGERED[11] event
TASKS_TRIGGER[12]	0x030		Trigger 12 for triggering the corresponding TRIGGERED[12] event
TASKS_TRIGGER[13]	0x034		Trigger 13 for triggering the corresponding TRIGGERED[13] event
TASKS_TRIGGER[14]	0x038		Trigger 14 for triggering the corresponding TRIGGERED[14] event
TASKS_TRIGGER[15]	0x03C		Trigger 15 for triggering the corresponding TRIGGERED[15] event
SUBSCRIBE_TRIGGER[0]	0x080		Subscribe configuration for task TRIGGER[0]
SUBSCRIBE_TRIGGER[1]	0x084		Subscribe configuration for task TRIGGER[1]
SUBSCRIBE_TRIGGER[2]	0x088		Subscribe configuration for task TRIGGER[2]
SUBSCRIBE_TRIGGER[3]	0x08C		Subscribe configuration for task TRIGGER[3]
SUBSCRIBE_TRIGGER[4]	0x090		Subscribe configuration for task TRIGGER[4]
SUBSCRIBE_TRIGGER[5]	0x094		Subscribe configuration for task TRIGGER[5]
SUBSCRIBE_TRIGGER[6]	0x098		Subscribe configuration for task TRIGGER[6]
SUBSCRIBE_TRIGGER[7]	0x09C		Subscribe configuration for task TRIGGER[7]
SUBSCRIBE_TRIGGER[8]	0x0A0		Subscribe configuration for task TRIGGER[8]
SUBSCRIBE_TRIGGER[9]	0x0A4		Subscribe configuration for task TRIGGER[9]
SUBSCRIBE_TRIGGER[10]	0x0A8		Subscribe configuration for task TRIGGER[10]
SUBSCRIBE_TRIGGER[11]	0x0AC		Subscribe configuration for task TRIGGER[11]
SUBSCRIBE_TRIGGER[12]	0x0B0		Subscribe configuration for task TRIGGER[12]
SUBSCRIBE_TRIGGER[13]	0x0B4		Subscribe configuration for task TRIGGER[13]
SUBSCRIBE_TRIGGER[14]	0x0B8		Subscribe configuration for task TRIGGER[14]
SUBSCRIBE_TRIGGER[15]	0x0BC		Subscribe configuration for task TRIGGER[15]
EVENTS_TRIGGERED[0]	0x100		Event number 0 generated by triggering the corresponding TRIGGER[0] task
EVENTS_TRIGGERED[1]	0x104		Event number 1 generated by triggering the corresponding TRIGGER[1] task
EVENTS_TRIGGERED[2]	0x108		Event number 2 generated by triggering the corresponding TRIGGER[2] task
EVENTS_TRIGGERED[3]	0x10C		Event number 3 generated by triggering the corresponding TRIGGER[3] task
EVENTS_TRIGGERED[4]	0x110		Event number 4 generated by triggering the corresponding TRIGGER[4] task
EVENTS_TRIGGERED[5]	0x114		Event number 5 generated by triggering the corresponding TRIGGER[5] task
EVENTS_TRIGGERED[6]	0x118		Event number 6 generated by triggering the corresponding TRIGGER[6] task
EVENTS_TRIGGERED[7]	0x11C		Event number 7 generated by triggering the corresponding TRIGGER[7] task
EVENTS_TRIGGERED[8]	0x120		Event number 8 generated by triggering the corresponding TRIGGER[8] task
EVENTS_TRIGGERED[9]	0x124		Event number 9 generated by triggering the corresponding TRIGGER[9] task
EVENTS_TRIGGERED[10]	0x128		Event number 10 generated by triggering the corresponding TRIGGER[10] task
EVENTS_TRIGGERED[11]	0x12C		Event number 11 generated by triggering the corresponding TRIGGER[11] task
EVENTS_TRIGGERED[12]	0x130		Event number 12 generated by triggering the corresponding TRIGGER[12] task
EVENTS_TRIGGERED[13]	0x134		Event number 13 generated by triggering the corresponding TRIGGER[13] task
EVENTS_TRIGGERED[14]	0x138		Event number 14 generated by triggering the corresponding TRIGGER[14] task
EVENTS_TRIGGERED[15]	0x13C		Event number 15 generated by triggering the corresponding TRIGGER[15] task
PUBLISH_TRIGGERED[0]	0x180		Publish configuration for event TRIGGERED[0]
PUBLISH_TRIGGERED[1]	0x184		Publish configuration for event TRIGGERED[1]
PUBLISH_TRIGGERED[2]	0x188		Publish configuration for event TRIGGERED[2]
PUBLISH_TRIGGERED[3]	0x18C		Publish configuration for event TRIGGERED[3]
PUBLISH_TRIGGERED[4]	0x190		Publish configuration for event TRIGGERED[4]
PUBLISH_TRIGGERED[5]	0x194		Publish configuration for event TRIGGERED[5]
PUBLISH_TRIGGERED[6]	0x198		Publish configuration for event TRIGGERED[6]
PUBLISH_TRIGGERED[7]	0x19C		Publish configuration for event TRIGGERED[7]
PUBLISH_TRIGGERED[8]	0x1A0		Publish configuration for event TRIGGERED[8]
PUBLISH_TRIGGERED[9]	0x1A4		Publish configuration for event TRIGGERED[9]
PUBLISH_TRIGGERED[10]	0x1A8		Publish configuration for event TRIGGERED[10]
PUBLISH_TRIGGERED[11]	0x1AC		Publish configuration for event TRIGGERED[11]
PUBLISH_TRIGGERED[12]	0x1B0		Publish configuration for event TRIGGERED[12]

Register	Offset	Security	Description
PUBLISH_TRIGGERED[13]	0x1B4		Publish configuration for event TRIGGERED[13]
PUBLISH_TRIGGERED[14]	0x1B8		Publish configuration for event TRIGGERED[14]
PUBLISH_TRIGGERED[15]	0x1BC		Publish configuration for event TRIGGERED[15]
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt

Table 33: Register overview

6.3.1.1 TASKS_TRIGGER[n] (n=0..15)

Address offset: 0x000 + (n × 0x4)

Trigger n for triggering the corresponding TRIGGERED[n] event

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_TRIGGER			Trigger n for triggering the corresponding TRIGGERED[n] event
			Trigger	1	Trigger task

6.3.1.2 SUBSCRIBE_TRIGGER[n] (n=0..15)

Address offset: 0x080 + (n × 0x4)

Subscribe configuration for task TRIGGER[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task TRIGGER[n] will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.3.1.3 EVENTS_TRIGGERED[n] (n=0..15)

Address offset: 0x100 + (n × 0x4)

Event number n generated by triggering the corresponding TRIGGER[n] task

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_TRIGGERED			Event number n generated by triggering the corresponding TRIGGER[n] task
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.3.1.4 PUBLISH_TRIGGERED[n] (n=0..15)

Address offset: 0x180 + (n × 0x4)

Publish configuration for event TRIGGERED[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event TRIGGERED[n] will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.3.1.5 INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-P	RW	TRIGGERED[i] (i=0..15)			Enable or disable interrupt for event TRIGGERED[i]
			Disabled	0	Disable
			Enabled	1	Enable

6.3.1.6 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-P	RW	TRIGGERED[i] (i=0..15)			Write '1' to enable interrupt for event TRIGGERED[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.3.1.7 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																			P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000			**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-P	RW	TRIGGERED[i] (i=0..15)			Write '1' to disable interrupt for event TRIGGERED[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.3.2 Electrical specification

6.3.2.1 EGU Electrical Specification

Symbol	Description	Min.	Typ.	Max.	Units
$t_{EGU,EVT}$	Latency between setting an EGU event flag and the system setting an interrupt		1		cycles

6.4 GPIO — General purpose input/output

The general purpose input/output pins (GPIOs) are grouped as one or more ports with each port having up to 32 GPIOs.

The number of ports and GPIOs per port might vary with product variant and package. Refer to Registers on page 95 and Pin assignments on page 379 for more information about the number of GPIOs that are supported.

GPIO has the following user-configurable features:

- Up to 32 GPIO pins per GPIO port
- Configurable output drive strength
- Internal pull-up and pull-down resistors
- Wake-up from high or low level triggers on all pins
- Trigger interrupt on state changes on any pin
- All pins can be used by the PPI task/event system
- One or more GPIO outputs can be controlled through PPI and GPIOTE channels
- All pins can be individually mapped to interface blocks for layout flexibility
- GPIO state changes captured on SENSE signal can be stored by LATCH register
- Support for Secure and Non-Secure attributes for pins in conjunction with the System Protection Unit (SPU - System protection unit on page 257)

The GPIO port peripheral implements up to 32 pins, `PIN0` through `PIN31`. Each of these pins can be individually configured in the PIN_CNF[n] registers (n=0..31).

The following parameters can be configured through these registers:

- Direction
- Drive strength
- Enabling of pull-up and pull-down resistors
- Pin sensing
- Input buffer disconnect
- Analog input (for selected pins)

The PIN_CNF registers are retained registers. See POWER — Power control on page 58 chapter for more information about retained registers.

6.4.1 Pin configuration

Pins can be individually configured, through the SENSE field in the PIN_CNF[n] register, to detect either a high level or a low level on their input.

When the correct level is detected on any such configured pin, the sense mechanism will set the DETECT signal high. Each pin has a separate DETECT signal. Default behavior, defined by the DETECTMODE register, is that the DETECT signals from all pins in the GPIO port are combined into one common DETECT signal that is routed throughout the system, which then can be utilized by other peripherals. This mechanism is functional in both System ON mode and System OFF mode. See GPIO port and the GPIO pin details on page 92.

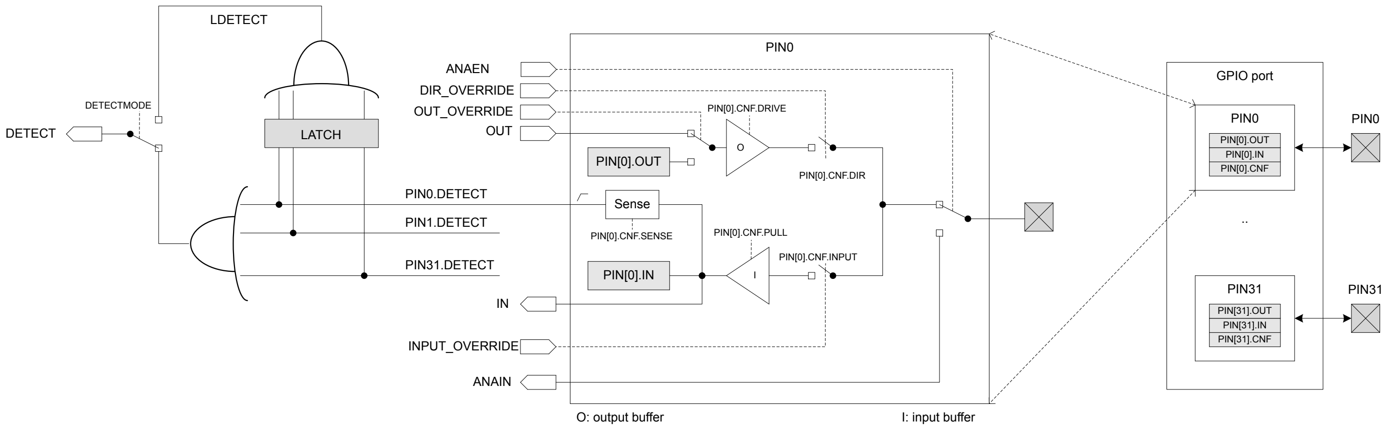


Figure 15: GPIO port and the GPIO pin details

In systems implementing a System Protection Unit, an extra DETECTMODE_SEC register is available to control the behaviour associated to pin marked as Secure, while the DETECTMODE register is restricted to pin marked as Non-Secure. Please refer to GPIO security on page 94 for more details.

GPIO port and the GPIO pin details on page 92 illustrates the GPIO port containing 32 individual pins, where `PIN0` is illustrated in more detail as a reference. All signals on the left side in the illustration are used by other peripherals in the system and therefore not directly available to the CPU.

Make sure that a pin is in a level that cannot trigger the sense mechanism before enabling it. The DETECT signal will go high immediately if the SENSE condition configured in the PIN_CNF registers is met when the sense mechanism is enabled. This will trigger a PORT event if the DETECT signal was low before enabling the sense mechanism. See GPIOTE — GPIO tasks and events on page 101.

See the following peripherals for more information about how the DETECT signal is used:

- POWER: uses the DETECT signal to exit from System OFF mode.
- GPIOTE: uses the DETECT signal to generate the PORT event. (GPIOTE_SEC is used for PORT event related to secure pins)

When a pin's PINx.DETECT signal goes high, a flag will be set in the LATCH register. For example, when the PIN0.DETECT signal goes high, bit 0 in the LATCH register will be set to '1'. If the CPU performs a clear operation on a bit in the LATCH register when the associated PINx.DETECT signal is high, the bit in the LATCH register will not be cleared. The LATCH register will only be cleared if the CPU explicitly clears it by writing a '1' to the bit that shall be cleared, i.e. the LATCH register will not be affected by a PINx.DETECT signal being set low.

The LDETECT signal will be set high when one or more bits in the LATCH register are '1'. The LDETECT signal will be set low when all bits in the LATCH register are successfully cleared to '0'.

If one or more bits in the LATCH register are '1' after the CPU has performed a clear operation on the LATCH registers, a rising edge will be generated on the LDETECT signal. This is illustrated in DETECT signal behavior on page 93.

Important: The CPU can read the LATCH register at any time to check if a SENSE condition has been met on one or more of the the GPIO pins, even if that condition is no longer met at the time the CPU queries the LATCH register. This mechanism will work even if the LDETECT signal is not used as the DETECT signal.

The LDETECT signal is by default not connected to the GPIO port's DETECT signal, but via the DETECTMODE register it is possible to change from default behavior to DETECT signal being derived directly from the LDETECT signal instead. See GPIO port and the GPIO pin details on page 92. DETECT signal behavior on page 93 illustrates the DETECT signal behavior for these two alternatives.

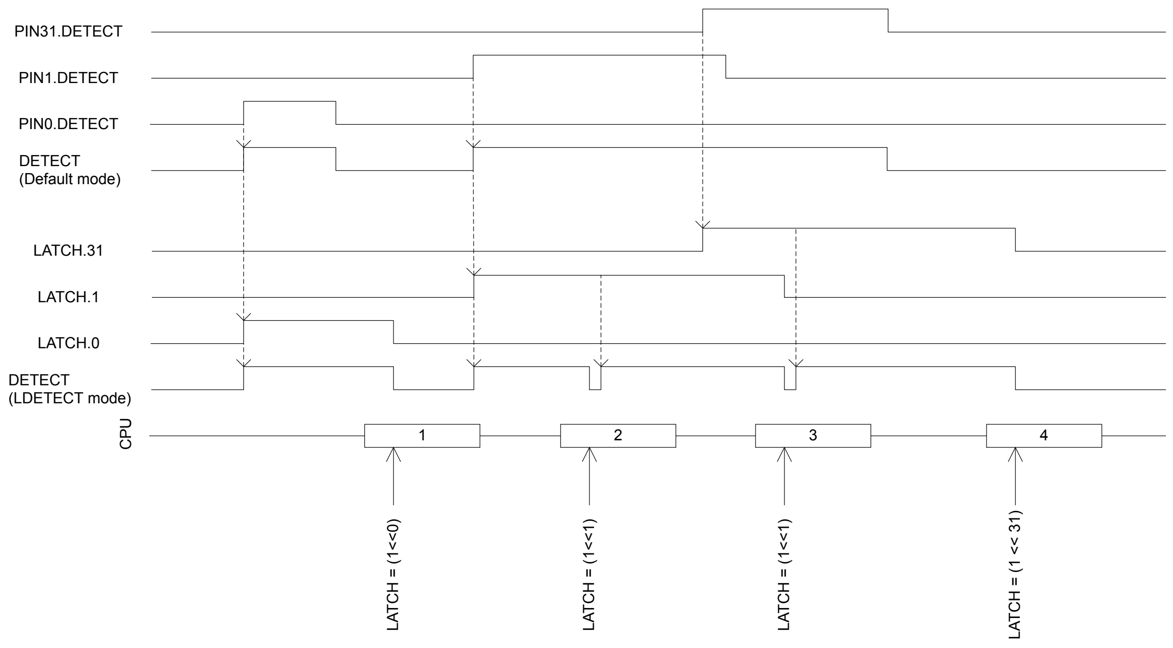


Figure 16: DETECT signal behavior

The input buffer of a GPIO pin can be disconnected from the pin to enable power savings when the pin is not used as an input, see GPIO port and the GPIO pin details on page 92. Inputs must be connected to get a valid input value in the IN register, and for the sense mechanism to get access to the pin.

Other peripherals in the system can connect to GPIO pins and override their output value and configuration, or read their analog or digital input value. See GPIO port and the GPIO pin details on page 92.

Selected pins also support analog input signals, see ANAIN in GPIO port and the GPIO pin details on page 92. The assignment of the analog pins can be found in Pin assignments on page 379.

The following delays should be taken into considerations:

- There is 2 CPU clock cycles delay from the GPIO pad to the GPIO.IN register
- The GPIO pad must be low (or high depending on the SENSE polarity) for 3 CPU clock cycles after DETECT has gone high in order to generate a new DETECT signal

Important: When a pin is configured as digital input, care has been taken to minimize increased current consumption when the input voltage is between V_{IL} and V_{IH}. However, it is a good practice to ensure that the external circuitry does not drive that pin to levels between V_{IL} and V_{IH} for a long period of time.

6.4.2 GPIO security

The General purpose input/output peripheral (GPIO) is implemented as a "split-security" peripheral. If marked as non-secure, it can be access by both secure and non-secure accesses but will behave differently depending of the access type :

A non-secure peripheral access will only be able to configure and control pins defined as non-secure in the System Protection Unit (SPU) GPIOPORT.PERM[] register(s).

A non-secure access to a register or a bitfield controlling a pin marked as secure in GPIO.PERM[] register(s) will be ignored:

- write accesses will have no effect
- read accesses will always return a zero value

No exception is triggered when a non-secure access targets a register or bitfield controlling a secure pin.

For example, if the bit *i* is set in the SPU.GPIO.PERM[0] register (declaring Pin P0.*i* as secure), then

- non-secure write accesses to OUT, OUTSET, OUTCLR, DIR, DIRSET, DIRCLR and LATCH registers will not be able to write to bit *i* of those registers
- non-secure write accesses to registers PIN[*i*].OUT and PIN_CNF[*i*] will be ignored
- non-secure read accesses to registers OUT, OUTSET, OUTCLR, IN, DIR, DIRSET, DIRCLR and LATCH will always read a 0 for the bit at position *i*
- non-secure read accesses to registers PIN[*i*].OUT, PIN[*i*].OUT and PIN_CNF[*i*] will always return 0

The GPIO.DETECTMODE and GPIO.DETECTMODE_SEC registers are handled differently than the other registers mentioned before. When accessed by a secure access, the DETECTMODE_SEC register control the source for the DETECT_SEC signal for the pins marked as secure. When accessed by a non-secure access, the DETECTMODE_SEC is read as zero and write accesses are ignored. The GPIO.DETECTMODE register controls the source for the DETECT_NSEC signal for the pins defined as non-secure.

The DETECT_NSEC signal is routed to the GPIOTE peripheral, allowing generation of events and interrupts from pins marked as non-secure. The DETECT_SEC signal is routed to the GPIOTESEC peripheral, allowing generation of events and interrupts from pins marked as secure. Principle of direct pin access on page 95 illustrates how the DETECT_NSEC and DETECT_SEC signals are generated from the GPIO PIN[].DETECT signals.

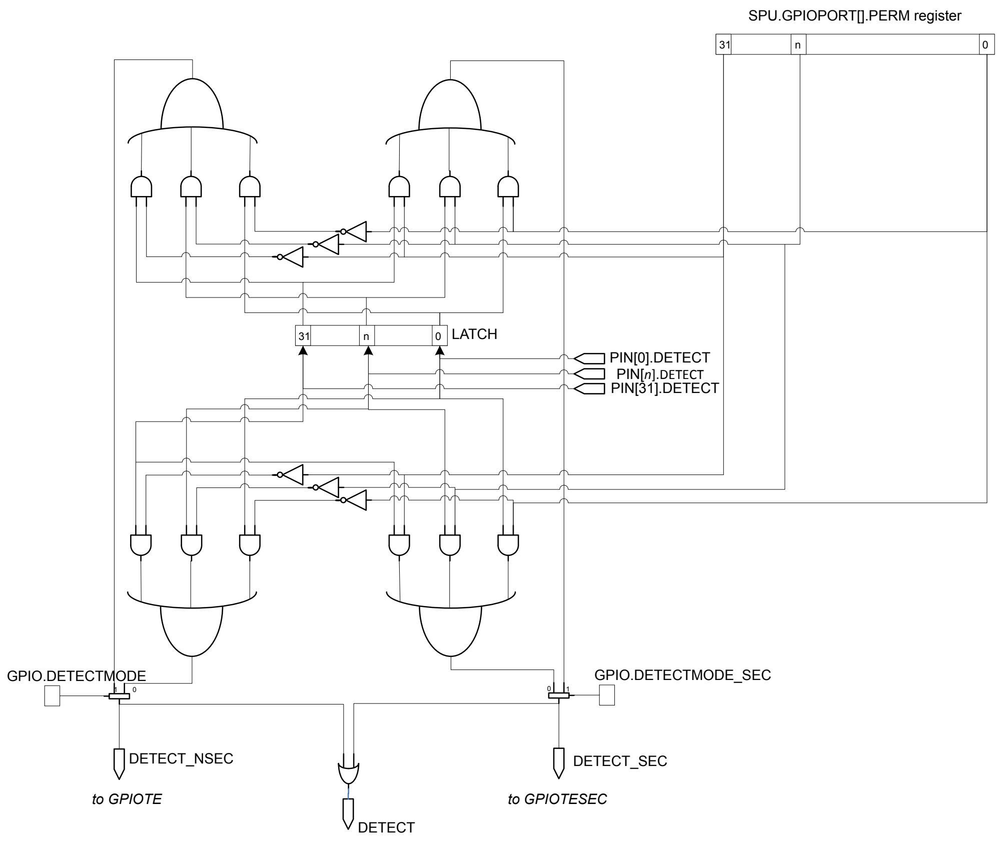


Figure 17: Principle of direct pin access

6.4.3 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50842500 0x40842500	GPIO	P0 : S P0 : NS	SPLIT	NA	General purpose input and output	

Table 34: Instances

Register	Offset	Security	Description
OUT	0x004		Write GPIO port
OUTSET	0x008		Set individual bits in GPIO port
OUTCLR	0x00C		Clear individual bits in GPIO port
IN	0x010		Read GPIO port
DIR	0x014		Direction of GPIO pins
DIRSET	0x018		DIR set register
DIRCLR	0x01C		DIR clear register
LATCH	0x020		Latch register indicating what GPIO pins that have met the criteria set in the PIN_CNF[n].SENSE registers
DETECTMODE	0x024		Select between default DETECT signal behaviour and LDETECT mode (For non-secure pin only)
DETECTMODE_SEC	0x028		Select between default DETECT signal behaviour and LDETECT mode (For secure pin only)
PIN_CNF[0]	0x200		Configuration of GPIO pins
PIN_CNF[1]	0x204		Configuration of GPIO pins

Register	Offset	Security	Description
PIN_CNF[2]	0x208		Configuration of GPIO pins
PIN_CNF[3]	0x20C		Configuration of GPIO pins
PIN_CNF[4]	0x210		Configuration of GPIO pins
PIN_CNF[5]	0x214		Configuration of GPIO pins
PIN_CNF[6]	0x218		Configuration of GPIO pins
PIN_CNF[7]	0x21C		Configuration of GPIO pins
PIN_CNF[8]	0x220		Configuration of GPIO pins
PIN_CNF[9]	0x224		Configuration of GPIO pins
PIN_CNF[10]	0x228		Configuration of GPIO pins
PIN_CNF[11]	0x22C		Configuration of GPIO pins
PIN_CNF[12]	0x230		Configuration of GPIO pins
PIN_CNF[13]	0x234		Configuration of GPIO pins
PIN_CNF[14]	0x238		Configuration of GPIO pins
PIN_CNF[15]	0x23C		Configuration of GPIO pins
PIN_CNF[16]	0x240		Configuration of GPIO pins
PIN_CNF[17]	0x244		Configuration of GPIO pins
PIN_CNF[18]	0x248		Configuration of GPIO pins
PIN_CNF[19]	0x24C		Configuration of GPIO pins
PIN_CNF[20]	0x250		Configuration of GPIO pins
PIN_CNF[21]	0x254		Configuration of GPIO pins
PIN_CNF[22]	0x258		Configuration of GPIO pins
PIN_CNF[23]	0x25C		Configuration of GPIO pins
PIN_CNF[24]	0x260		Configuration of GPIO pins
PIN_CNF[25]	0x264		Configuration of GPIO pins
PIN_CNF[26]	0x268		Configuration of GPIO pins
PIN_CNF[27]	0x26C		Configuration of GPIO pins
PIN_CNF[28]	0x270		Configuration of GPIO pins
PIN_CNF[29]	0x274		Configuration of GPIO pins
PIN_CNF[30]	0x278		Configuration of GPIO pins
PIN_CNF[31]	0x27C		Configuration of GPIO pins

Table 35: Register overview

6.4.3.1 OUT

Address offset: 0x004

Write GPIO port

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Pin i
			Low	0	Pin driver is low
			High	1	Pin driver is high

6.4.3.2 OUTSET

Address offset: 0x008

Set individual bits in GPIO port

Read: reads value of OUT register.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Pin i
			Low	0	Read: pin driver is low
			High	1	Read: pin driver is high
			Set	1	Write: writing a '1' sets the pin high; writing a '0' has no effect

6.4.3.3 OUTCLR

Address offset: 0x00C

Clear individual bits in GPIO port

Read: reads value of OUT register.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Pin i
			Low	0	Read: pin driver is low
			High	1	Read: pin driver is high
			Clear	1	Write: writing a '1' sets the pin low; writing a '0' has no effect

6.4.3.4 IN

Address offset: 0x010

Read GPIO port

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-f	R	PIN[i] (i=0..31)			Pin i
			Low	0	Pin input is low
			High	1	Pin input is high

6.4.3.5 DIR

Address offset: 0x014

Direction of GPIO pins

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Pin i
			Input	0	Pin set as input
			Output	1	Pin set as output

6.4.3.6 DIRSET

Address offset: 0x018

DIR set register

Read: reads value of DIR register.

Bit number				31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID				f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000				**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Set as output pin i
			Input	0	Read: pin set as input
			Output	1	Read: pin set as output
			Set	1	Write: writing a '1' sets pin to output; writing a '0' has no effect

6.4.3.7 DIRCLR

Address offset: 0x01C

DIR clear register

Read: reads value of DIR register.

Bit number				31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID				f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000				**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Set as input pin i
			Input	0	Read: pin set as input
			Output	1	Read: pin set as output
			Clear	1	Write: writing a '1' sets pin to input; writing a '0' has no effect

6.4.3.8 LATCH

Address offset: 0x020

Latch register indicating what GPIO pins that have met the criteria set in the PIN_CNF[n].SENSE registers

Bit number				31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID				f	e	d	c	b	a	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Reset 0x00000000				**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-f	RW	PIN[i] (i=0..31)			Status on whether PINi has met criteria set in PIN_CNFi.SENSE register. Write '1' to clear.
			NotLatched	0	Criteria has not been met
			Latched	1	Criteria has been met

6.4.3.9 DETECTMODE

Address offset: 0x024

Select between default DETECT signal behaviour and LDETECT mode (For non-secure pin only)

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	DETECTMODE			Select between default DETECT signal behaviour and LDETECT mode
			Default	0	DETECT directly connected to PIN DETECT signals
			LDETECT	1	Use the latched LDETECT behaviour

6.4.3.10 DETECTMODE_SEC

Address offset: 0x028

Select between default DETECT signal behaviour and LDETECT mode (For secure pin only)

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	DETECTMODE			Select between default DETECT signal behaviour and LDETECT mode
			Default	0	DETECT directly connected to PIN DETECT signals
			LDETECT	1	Use the latched LDETECT behaviour

6.4.3.11 PIN_CNF[n] (n=0..31)

Address offset: 0x200 + (n × 0x4)

Configuration of GPIO pins

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID															E	E						D	D	D					C	C	B	A
Reset 0x00000002	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	DIR			Pin direction. Same physical register as DIR register
			Input	0	Configure pin as an input pin
			Output	1	Configure pin as an output pin
B	RW	INPUT			Connect or disconnect input buffer
			Connect	0	Connect input buffer
			Disconnect	1	Disconnect input buffer
C	RW	PULL			Pull configuration
			Disabled	0	No pull
			Pulldown	1	Pull down on pin
			Pullup	3	Pull up on pin
D	RW	DRIVE			Drive configuration
			S0S1	0	Standard '0', standard '1'
			H0S1	1	High drive '0', standard '1'
			S0H1	2	Standard '0', high drive '1'
			H0H1	3	High drive '0', high 'drive '1''
			D0S1	4	Disconnect '0' standard '1' (normally used for wired-or connections)
			D0H1	5	Disconnect '0', high drive '1' (normally used for wired-or connections)
			S0D1	6	Standard '0'. disconnect '1' (normally used for wired-and connections)

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID															E	E						D	D	D					C	C	B	A
Reset 0x00000002	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**

ID	Acce	Field	Value ID	Value	Description
			H0D1	7	High drive '0', disconnect '1' (normally used for wired-and connections)
E	RW	SENSE			Pin sensing mechanism
			Disabled	0	Disabled
			High	2	Sense for high level
			Low	3	Sense for low level

6.4.4 Electrical specification

6.4.4.1 GPIO Electrical Specification

Symbol	Description	Min.	Typ.	Max.	Units
V_{IH}	Input high voltage	0.7 x VDD		VDD	V
V_{IL}	Input low voltage	VSS		0.3 x VDD	V
$V_{OH,SD}$	Output high voltage, standard drive, 0.5 mA, VDD ≥1.7	VDD-0.4		VDD	V
$V_{OH,HDH}$	Output high voltage, high drive, 5 mA, VDD >= 2.7 V	VDD-0.4		VDD	V
$V_{OH,HDL}$	Output high voltage, high drive, 3 mA, VDD >= 1.7 V	VDD-0.4		VDD	V
$V_{OL,SD}$	Output low voltage, standard drive, 0.5 mA, VDD ≥1.7	VSS		VSS+0.4	V
$V_{OL,HDH}$	Output low voltage, high drive, 5 mA, VDD >= 2.7 V	VSS		VSS+0.4	V
$V_{OL,HDL}$	Output low voltage, high drive, 3 mA, VDD >= 1.7 V	VSS		VSS+0.4	V
$I_{OL,SD}$	Current at VSS+0.4 V, output set low, standard drive, VDD ≥1.7	1	2	4	mA
$I_{OL,HDH}$	Current at VSS+0.4 V, output set low, high drive, VDD >= 2.7 V	6	10	15	mA
$I_{OL,HDL}$	Current at VSS+0.4 V, output set low, high drive, VDD >= 1.7 V	3			mA
$I_{OH,SD}$	Current at VDD-0.4 V, output set high, standard drive, VDD ≥1.7	1	2	4	mA
$I_{OH,HDH}$	Current at VDD-0.4 V, output set high, high drive, VDD >= 2.7 V	6	9	14	mA
$I_{OH,HDL}$	Current at VDD-0.4 V, output set high, high drive, VDD >= 1.7 V	3			mA
$t_{RF,15pF}$	Rise/fall time, standard drive mode, 10-90%, 15 pF load[1]	6	9	19	ns
$t_{RF,25pF}$	Rise/fall time, standard drive mode, 10-90%, 25 pF load[1]	10	13	30	ns
$t_{RF,50pF}$	Rise/fall time, standard drive mode, 10-90%, 50 pF load[1]	18	25	61	ns
$t_{HRF,15pF}$	Rise/Fall time, high drive mode, 10-90%, 15 pF load[1]	2	4	8	ns
$t_{HRF,25pF}$	Rise/Fall time, high drive mode, 10-90%, 25 pF load[1]	3	5	11	ns
$t_{HRF,50pF}$	Rise/Fall time, high drive mode, 10-90%, 50 pF load[1]	5	8	19	ns
R_{PU}	Pull-up resistance	11	13	16	kΩ
R_{PD}	Pull-down resistance	11	13	16	kΩ
C_{PAD}	Pad capacitance		3		pF

[1] Rise and fall times based on simulations

6.5 GPIOTE — GPIO tasks and events

The GPIO tasks and events (GPIOTE) module provides functionality for accessing GPIO pins using tasks and events. Each GPIOTE channel can be assigned to one pin.

A GPIOTE block enables GPIOs to generate events on pin state change which can be used to carry out tasks through the PPI system. A GPIO can also be driven to change state on system events using the PPI system. Low power detection of pin state changes is possible when in System ON or System OFF.

Instance	Number of GPIOTE channels
GPIOTE	8

Table 36: GPIOTE properties

Up to three tasks can be used in each GPIOTE channel for performing write operations to a pin. Two tasks are fixed (SET and CLR), and one (OUT) is configurable to perform following operations:

- Set
- Clear
- Toggle

An event can be generated in each GPIOTE channel from one of the following input conditions:

- Rising edge
- Falling edge
- Any change

6.5.1 Pin events and tasks

The GPIOTE module has a number of tasks and events that can be configured to operate on individual GPIO pins.

The tasks (SET[n], CLR[n] and OUT[n]) can be used for writing to individual pins, and the events (IN[n]) can be generated from changes occurring at the inputs of individual pins.

The SET task will set the pin selected in GPIOTE.CONFIG[n].PSEL to high.

The CLR task will set the pin low.

The effect of the OUT task on the pin is configurable in CONFIG[n].POLARITY , and can either set the pin high, set it low, or toggle it.

The tasks and events are configured using the CONFIG[n] registers. Every set of SET, CLR and OUT[n] tasks and IN[n] events has one CONFIG[n] register associated with it.

As long as a SET[n], CLR[n] and OUT[n] task or an IN[n] event is configured to control a pin **n**, the pin's output value will only be updated by the GPIOTE module. The pin's output value as specified in the GPIO will therefore be ignored as long as the pin is controlled by GPIOTE. Attempting to write a pin as a normal GPIO pin will have no effect. When the GPIOTE is disconnected from a pin, see MODE field in CONFIG[n] register, the associated pin will get the output and configuration values specified in the GPIO module.

When conflicting tasks are triggered simultaneously (i.e. during the same clock cycle) in one channel, the precedence of the tasks will be as described in Task priorities on page 102.

Priority	Task
1	OUT
2	CLR
3	SET

Table 37: Task priorities

When setting the CONFIG[n] registers, MODE=Disabled does not have the same effect as MODE=Task and POLARITY=None. In the latter case, a CLR or SET task occurring at the exact same time as OUT will end up with no change on the pin, according to the priorities described in the table above.

When a GPIOTE channel is configured to operate on a pin as a task, the initial value of that pin is configured in the OUTINIT field of CONFIG[n].

6.5.2 Port event

PORT is an event that can be generated from multiple input pins using the GPIO DETECT signal.

The event will be generated on the rising edge of the DETECT signal. See GPIO — General purpose input/output on page 91 for more information about the DETECT signal.

Putting the system into System ON IDLE while DETECT is high will not cause DETECT to wake the system up again. Make sure to clear all DETECT sources before entering sleep. If the LATCH register is used as a source, if any bit in LATCH is still high after clearing all or part of the register (for instance due to one of the PINx.DETECT signal still high), a new rising edge will be generated on DETECT, see Pin configuration on page 92.

Trying to put the system to System OFF while DETECT is high will cause a wakeup from System OFF reset.

This feature is always enabled although the peripheral itself appears to be IDLE, that is, no clocks or other power intensive infrastructure have to be requested to keep this feature enabled. This feature can therefore be used to wake up the CPU from a WFI or WFE type sleep in System ON with all peripherals and the CPU idle, that is, lowest power consumption in System ON mode.

In order to prevent spurious interrupts from the PORT event while configuring the sources, the user shall first disable interrupts on the PORT event (through INTENCLR.PORT), then configure the sources (PIN_CNF[n].SENSE), clear any potential event that could have occurred during configuration (write '1' to EVENTS_PORT), and finally enable interrupts (through INTENSET.PORT).

6.5.3 Tasks and events pin configuration

Each GPIOTE channel is associated with one physical GPIO pin through the CONFIG.PSEL field.

When Event mode is selected in CONFIG.MODE, the pin specified by CONFIG.PSEL will be configured as an input, overriding the DIR setting in GPIO. Similarly, when Task mode is selected in CONFIG.MODE, the pin specified by CONFIG.PSEL will be configured as an output overriding the DIR setting and OUT value in GPIO. When Disabled is selected in CONFIG.MODE, the pin specified by CONFIG.PSEL will use its configuration from the PIN[n].CNF registers in GPIO.

Only one GPIOTE channel can be assigned to one physical pin. Failing to do so may result in unpredictable behavior.

6.5.4 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x5000D000	GPIOTE	GPIOTE0	S	NA	Secure GPIO tasks and events	
0x40031000	GPIOTE	GPIOTE1	NS	NA	Non Secure GPIO tasks and events	

Table 38: Instances

Register	Offset	Security	Description
TASKS_OUT[0]	0x000		Task for writing to pin specified in CONFIG[0].PSEL. Action on pin is configured in CONFIG[0].POLARITY.
TASKS_OUT[1]	0x004		Task for writing to pin specified in CONFIG[1].PSEL. Action on pin is configured in CONFIG[1].POLARITY.
TASKS_OUT[2]	0x008		Task for writing to pin specified in CONFIG[2].PSEL. Action on pin is configured in CONFIG[2].POLARITY.
TASKS_OUT[3]	0x00C		Task for writing to pin specified in CONFIG[3].PSEL. Action on pin is configured in CONFIG[3].POLARITY.
TASKS_OUT[4]	0x010		Task for writing to pin specified in CONFIG[4].PSEL. Action on pin is configured in CONFIG[4].POLARITY.
TASKS_OUT[5]	0x014		Task for writing to pin specified in CONFIG[5].PSEL. Action on pin is configured in CONFIG[5].POLARITY.
TASKS_OUT[6]	0x018		Task for writing to pin specified in CONFIG[6].PSEL. Action on pin is configured in CONFIG[6].POLARITY.
TASKS_OUT[7]	0x01C		Task for writing to pin specified in CONFIG[7].PSEL. Action on pin is configured in CONFIG[7].POLARITY.
TASKS_SET[0]	0x030		Task for writing to pin specified in CONFIG[0].PSEL. Action on pin is to set it high.
TASKS_SET[1]	0x034		Task for writing to pin specified in CONFIG[1].PSEL. Action on pin is to set it high.
TASKS_SET[2]	0x038		Task for writing to pin specified in CONFIG[2].PSEL. Action on pin is to set it high.
TASKS_SET[3]	0x03C		Task for writing to pin specified in CONFIG[3].PSEL. Action on pin is to set it high.
TASKS_SET[4]	0x040		Task for writing to pin specified in CONFIG[4].PSEL. Action on pin is to set it high.
TASKS_SET[5]	0x044		Task for writing to pin specified in CONFIG[5].PSEL. Action on pin is to set it high.
TASKS_SET[6]	0x048		Task for writing to pin specified in CONFIG[6].PSEL. Action on pin is to set it high.
TASKS_SET[7]	0x04C		Task for writing to pin specified in CONFIG[7].PSEL. Action on pin is to set it high.
TASKS_CLR[0]	0x060		Task for writing to pin specified in CONFIG[0].PSEL. Action on pin is to set it low.
TASKS_CLR[1]	0x064		Task for writing to pin specified in CONFIG[1].PSEL. Action on pin is to set it low.
TASKS_CLR[2]	0x068		Task for writing to pin specified in CONFIG[2].PSEL. Action on pin is to set it low.
TASKS_CLR[3]	0x06C		Task for writing to pin specified in CONFIG[3].PSEL. Action on pin is to set it low.
TASKS_CLR[4]	0x070		Task for writing to pin specified in CONFIG[4].PSEL. Action on pin is to set it low.
TASKS_CLR[5]	0x074		Task for writing to pin specified in CONFIG[5].PSEL. Action on pin is to set it low.
TASKS_CLR[6]	0x078		Task for writing to pin specified in CONFIG[6].PSEL. Action on pin is to set it low.
TASKS_CLR[7]	0x07C		Task for writing to pin specified in CONFIG[7].PSEL. Action on pin is to set it low.
SUBSCRIBE_OUT[0]	0x080		Subscribe configuration for task OUT[0]
SUBSCRIBE_OUT[1]	0x084		Subscribe configuration for task OUT[1]
SUBSCRIBE_OUT[2]	0x088		Subscribe configuration for task OUT[2]
SUBSCRIBE_OUT[3]	0x08C		Subscribe configuration for task OUT[3]
SUBSCRIBE_OUT[4]	0x090		Subscribe configuration for task OUT[4]
SUBSCRIBE_OUT[5]	0x094		Subscribe configuration for task OUT[5]
SUBSCRIBE_OUT[6]	0x098		Subscribe configuration for task OUT[6]
SUBSCRIBE_OUT[7]	0x09C		Subscribe configuration for task OUT[7]
SUBSCRIBE_SET[0]	0x0B0		Subscribe configuration for task SET[0]
SUBSCRIBE_SET[1]	0x0B4		Subscribe configuration for task SET[1]
SUBSCRIBE_SET[2]	0x0B8		Subscribe configuration for task SET[2]

Register	Offset	Security	Description
SUBSCRIBE_SET[3]	0x0BC		Subscribe configuration for task SET[3]
SUBSCRIBE_SET[4]	0x0C0		Subscribe configuration for task SET[4]
SUBSCRIBE_SET[5]	0x0C4		Subscribe configuration for task SET[5]
SUBSCRIBE_SET[6]	0x0C8		Subscribe configuration for task SET[6]
SUBSCRIBE_SET[7]	0x0CC		Subscribe configuration for task SET[7]
SUBSCRIBE_CLR[0]	0x0E0		Subscribe configuration for task CLR[0]
SUBSCRIBE_CLR[1]	0x0E4		Subscribe configuration for task CLR[1]
SUBSCRIBE_CLR[2]	0x0E8		Subscribe configuration for task CLR[2]
SUBSCRIBE_CLR[3]	0x0EC		Subscribe configuration for task CLR[3]
SUBSCRIBE_CLR[4]	0x0F0		Subscribe configuration for task CLR[4]
SUBSCRIBE_CLR[5]	0x0F4		Subscribe configuration for task CLR[5]
SUBSCRIBE_CLR[6]	0x0F8		Subscribe configuration for task CLR[6]
SUBSCRIBE_CLR[7]	0x0FC		Subscribe configuration for task CLR[7]
EVENTS_IN[0]	0x100		Event generated from pin specified in CONFIG[0].PSEL
EVENTS_IN[1]	0x104		Event generated from pin specified in CONFIG[1].PSEL
EVENTS_IN[2]	0x108		Event generated from pin specified in CONFIG[2].PSEL
EVENTS_IN[3]	0x10C		Event generated from pin specified in CONFIG[3].PSEL
EVENTS_IN[4]	0x110		Event generated from pin specified in CONFIG[4].PSEL
EVENTS_IN[5]	0x114		Event generated from pin specified in CONFIG[5].PSEL
EVENTS_IN[6]	0x118		Event generated from pin specified in CONFIG[6].PSEL
EVENTS_IN[7]	0x11C		Event generated from pin specified in CONFIG[7].PSEL
EVENTS_PORT	0x17C		Event generated from multiple input GPIO pins with SENSE mechanism enabled
PUBLISH_IN[0]	0x180		Publish configuration for event IN[0]
PUBLISH_IN[1]	0x184		Publish configuration for event IN[1]
PUBLISH_IN[2]	0x188		Publish configuration for event IN[2]
PUBLISH_IN[3]	0x18C		Publish configuration for event IN[3]
PUBLISH_IN[4]	0x190		Publish configuration for event IN[4]
PUBLISH_IN[5]	0x194		Publish configuration for event IN[5]
PUBLISH_IN[6]	0x198		Publish configuration for event IN[6]
PUBLISH_IN[7]	0x19C		Publish configuration for event IN[7]
PUBLISH_PORT	0x1FC		Publish configuration for event PORT
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
CONFIG[0]	0x510		Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event
CONFIG[1]	0x514		Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event
CONFIG[2]	0x518		Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event
CONFIG[3]	0x51C		Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event
CONFIG[4]	0x520		Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event
CONFIG[5]	0x524		Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event
CONFIG[6]	0x528		Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event
CONFIG[7]	0x52C		Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event

Table 39: Register overview

6.5.4.1 TASKS_OUT[n] (n=0..7)

Address offset: 0x000 + (n × 0x4)

Task for writing to pin specified in CONFIG[n].PSEL. Action on pin is configured in CONFIG[n].POLARITY.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_OUT			Task for writing to pin specified in CONFIG[n].PSEL. Action on pin is configured in CONFIG[n].POLARITY.
			Trigger	1	Trigger task

6.5.4.2 TASKS_SET[n] (n=0..7)

Address offset: 0x030 + (n × 0x4)

Task for writing to pin specified in CONFIG[n].PSEL. Action on pin is to set it high.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_SET			Task for writing to pin specified in CONFIG[n].PSEL. Action on pin is to set it high.
			Trigger	1	Trigger task

6.5.4.3 TASKS_CLR[n] (n=0..7)

Address offset: 0x060 + (n × 0x4)

Task for writing to pin specified in CONFIG[n].PSEL. Action on pin is to set it low.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_CLR			Task for writing to pin specified in CONFIG[n].PSEL. Action on pin is to set it low.
			Trigger	1	Trigger task

6.5.4.4 SUBSCRIBE_OUT[n] (n=0..7)

Address offset: 0x080 + (n × 0x4)

Subscribe configuration for task OUT[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task OUT[n] will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.5.4.5 SUBSCRIBE_SET[n] (n=0..7)

Address offset: 0x0B0 + (n × 0x4)

Subscribe configuration for task SET[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task SET[n] will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.5.4.6 SUBSCRIBE_CLR[n] (n=0..7)

Address offset: 0x0E0 + (n × 0x4)

Subscribe configuration for task CLR[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task CLR[n] will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.5.4.7 EVENTS_IN[n] (n=0..7)

Address offset: 0x100 + (n × 0x4)

Event generated from pin specified in CONFIG[n].PSEL

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_IN			Event generated from pin specified in CONFIG[n].PSEL
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.5.4.8 EVENTS_PORT

Address offset: 0x17C

Event generated from multiple input GPIO pins with SENSE mechanism enabled

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_PORT			Event generated from multiple input GPIO pins with SENSE mechanism enabled
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.5.4.9 PUBLISH_IN[n] (n=0..7)

Address offset: 0x180 + (n × 0x4)

Publish configuration for event IN[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event IN[n] will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.5.4.10 PUBLISH_PORT

Address offset: 0x1FC

Publish configuration for event PORT

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event PORT will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.5.4.11 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	I																								H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	RW	IN[i] (i=0..7)			Write '1' to enable interrupt for event IN[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
I	RW	PORT			Write '1' to enable interrupt for event PORT
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.5.4.12 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	I																								H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	RW	IN[i] (i=0..7)			Write '1' to disable interrupt for event IN[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
I	RW	PORT			Write '1' to disable interrupt for event PORT
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.5.4.13 CONFIG[n] (n=0..7)

Address offset: 0x510 + (n × 0x4)

Configuration for OUT[n], SET[n] and CLR[n] tasks and IN[n] event

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID												E			D	D				B	B	B	B	B							A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	MODE			Mode
			Disabled	0	Disabled. Pin specified by PSEL will not be acquired by the GPIOTE module.
			Event	1	Event mode The pin specified by PSEL will be configured as an input and the IN[n] event will be generated if operation specified in POLARITY occurs on the pin.
			Task	3	Task mode The GPIO specified by PSEL will be configured as an output and triggering the SET[n], CLR[n] or OUT[n] task will perform the operation specified by POLARITY on the pin. When enabled as a task the GPIOTE module will acquire the pin and the pin can no longer be written as a regular output pin from the GPIO module.
B	RW	PSEL		[0..31]	GPIO number associated with SET[n], CLR[n] and OUT[n] tasks and IN[n] event
D	RW	POLARITY			When In task mode: Operation to be performed on output when OUT[n] task is triggered. When In event mode: Operation on input that shall trigger IN[n] event.
			None	0	Task mode: No effect on pin from OUT[n] task. Event mode: no IN[n] event generated on pin activity.
			LoToHi	1	Task mode: Set pin from OUT[n] task. Event mode: Generate IN[n] event when rising edge on pin.
			HiToLo	2	Task mode: Clear pin from OUT[n] task. Event mode: Generate IN[n] event when falling edge on pin.
			Toggle	3	Task mode: Toggle pin from OUT[n]. Event mode: Generate IN[n] when any change on pin.
E	RW	OUTINIT			When in task mode: Initial value of the output when the GPIOTE channel is configured. When in event mode: No effect.
			Low	0	Task mode: Initial value of pin before task triggering is low

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID												E			D	D				B	B	B	B	B							A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce Field	Value ID	Value	Description
		High	1	Task mode: Initial value of pin before task triggering is high

6.5.5 Electrical specification

6.6 IPC — Inter-Processor Communication

The IPC peripheral is used to send and receive events between processors in the system.

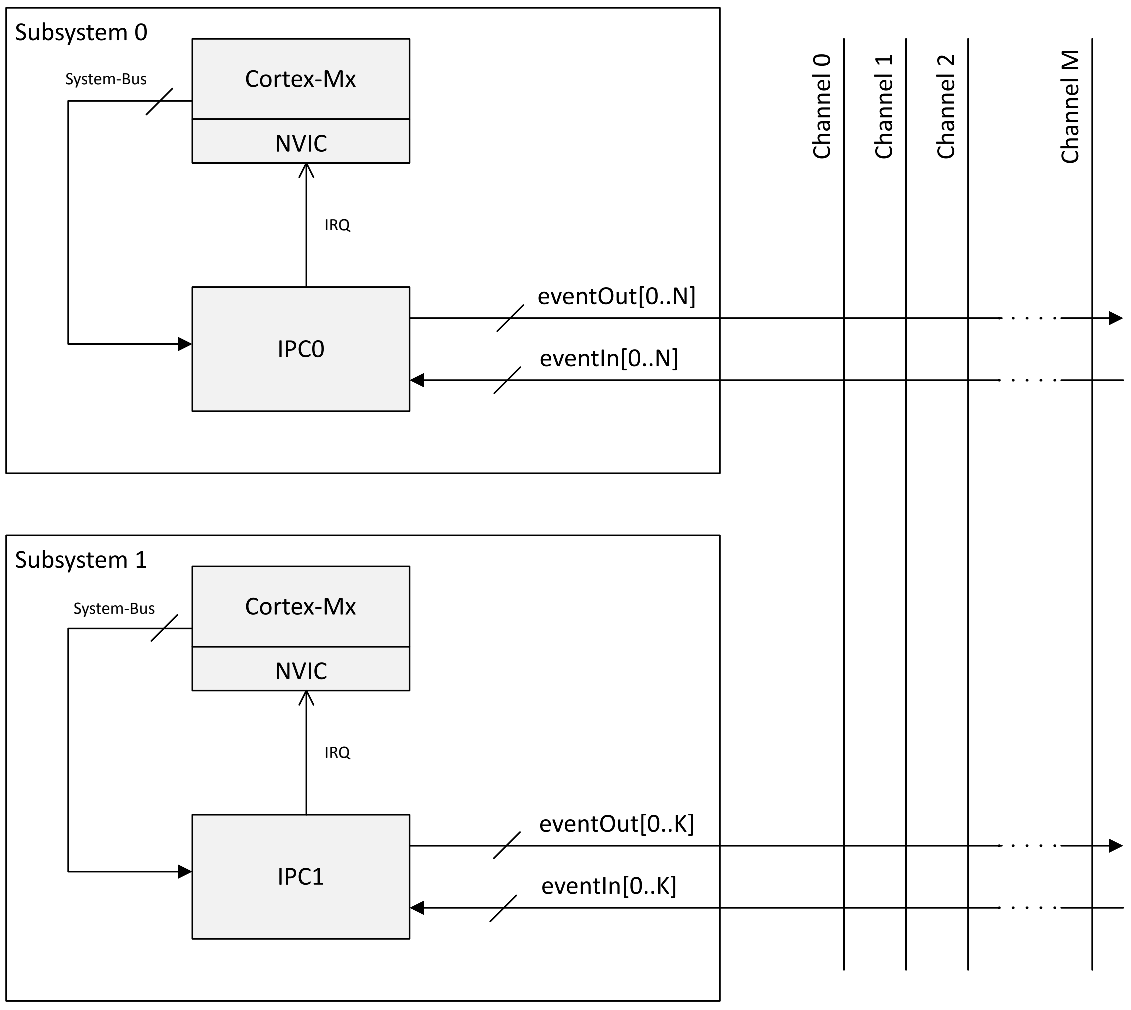


Figure 18: IPC block diagram

Functional description

IPC block diagram on page 109 illustrates the Inter-Processor Communication (IPC) peripheral. In a multi-core system, every CPU instance shall have one dedicated IPC peripheral. The IPC peripheral can be used to send and receive events to and from other IPC peripherals. An instance of the IPC peripheral can have #N send tasks and #N receive events. A single send task can be configured to signal an event on one or more channels, and a receive event can be configured to listen on one or more channels. The channels that are triggered in a send task can be configured through the SEND_CNF registers, and the channels that trigger a receive event is configured through the RECEIVE_CNF registers. A send task can be viewed as broadcasting events onto one or more channels, and a receive event can be seen as subscribing to a subset of channels. It is possible for multiple IPCs to trigger events onto the same channel at the same time, in this case it will look as a single event from the IPC subscriber.

The number of channels and send/receive events per IPC are implementation specific, and needs to be looked up in the reference manual for your specific device.

An event itself often does not contain any relevant information itself other than to signal that "something has occurred". Shared memory can be used to carry additional information between processors, e.g. in the form of command/event queues. It is up to software to assign a logical functionality to a channel. For instance one channel can be used to signal that a command is ready to be executed and any processor in the system can subscribe to that particular channel and decode/execute the command.

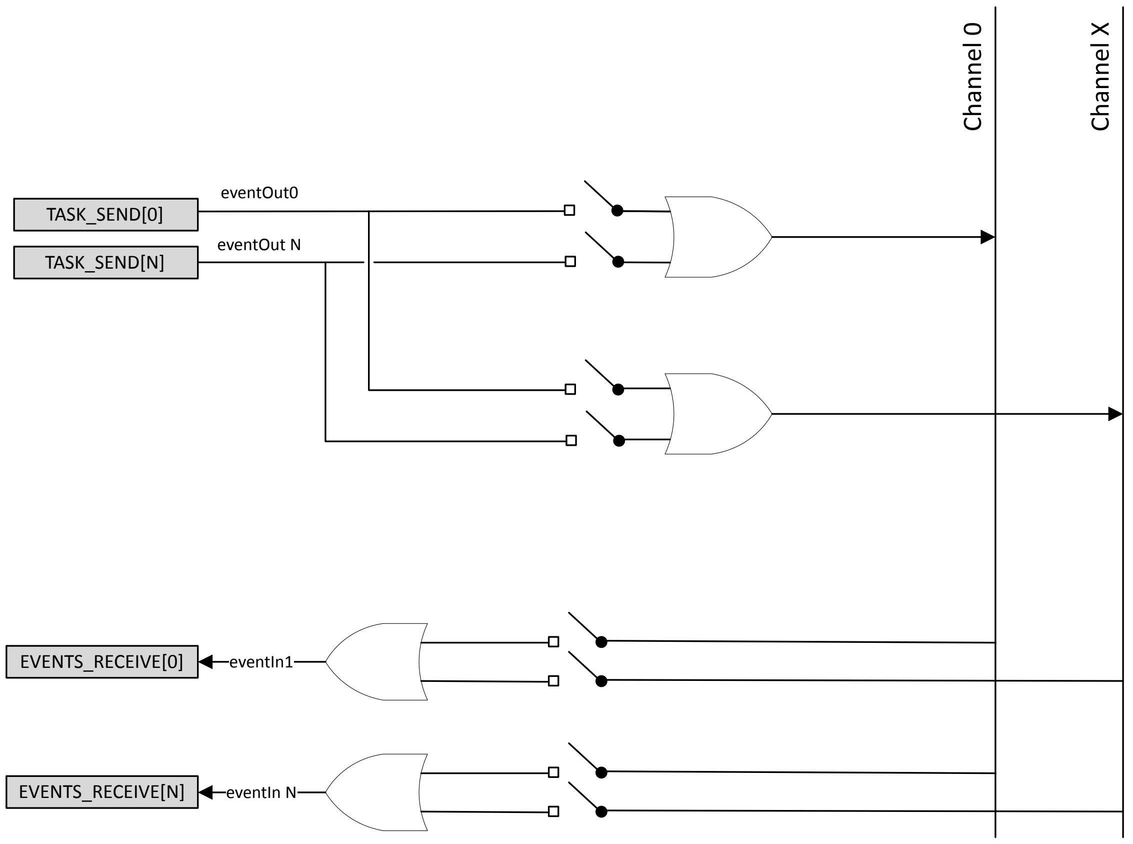


Figure 19: IPC SEND_CNF and RECEIVE_CNF

IPC SEND_CNF and RECEIVE_CNF on page 110 illustrates how the SEND_CNF and RECEIVE_CNF registers work. A send task be connected to all channels, and a receive event can be connected to all channels.

6.6.1 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x5002A000 0x4002A000	IPC	IPC : S IPC : NS	US	NA	Interprocessor communication	

Table 40: Instances

Register	Offset	Security	Description
TASKS_SEND[0]	0x000		Trigger events on channel enabled in SEND_CNF[0].
TASKS_SEND[1]	0x004		Trigger events on channel enabled in SEND_CNF[1].
TASKS_SEND[2]	0x008		Trigger events on channel enabled in SEND_CNF[2].
TASKS_SEND[3]	0x00C		Trigger events on channel enabled in SEND_CNF[3].

Register	Offset	Security	Description
TASKS_SEND[4]	0x010		Trigger events on channel enabled in SEND_CNF[4].
TASKS_SEND[5]	0x014		Trigger events on channel enabled in SEND_CNF[5].
TASKS_SEND[6]	0x018		Trigger events on channel enabled in SEND_CNF[6].
TASKS_SEND[7]	0x01C		Trigger events on channel enabled in SEND_CNF[7].
SUBSCRIBE_SEND[0]	0x080		Subscribe configuration for task SEND[0]
SUBSCRIBE_SEND[1]	0x084		Subscribe configuration for task SEND[1]
SUBSCRIBE_SEND[2]	0x088		Subscribe configuration for task SEND[2]
SUBSCRIBE_SEND[3]	0x08C		Subscribe configuration for task SEND[3]
SUBSCRIBE_SEND[4]	0x090		Subscribe configuration for task SEND[4]
SUBSCRIBE_SEND[5]	0x094		Subscribe configuration for task SEND[5]
SUBSCRIBE_SEND[6]	0x098		Subscribe configuration for task SEND[6]
SUBSCRIBE_SEND[7]	0x09C		Subscribe configuration for task SEND[7]
EVENTS_RECEIVE[0]	0x100		Event received on one or more of the enabled channels in RECEIVE_CNF[n].
EVENTS_RECEIVE[1]	0x104		Event received on one or more of the enabled channels in RECEIVE_CNF[n].
EVENTS_RECEIVE[2]	0x108		Event received on one or more of the enabled channels in RECEIVE_CNF[n].
EVENTS_RECEIVE[3]	0x10C		Event received on one or more of the enabled channels in RECEIVE_CNF[n].
EVENTS_RECEIVE[4]	0x110		Event received on one or more of the enabled channels in RECEIVE_CNF[n].
EVENTS_RECEIVE[5]	0x114		Event received on one or more of the enabled channels in RECEIVE_CNF[n].
EVENTS_RECEIVE[6]	0x118		Event received on one or more of the enabled channels in RECEIVE_CNF[n].
EVENTS_RECEIVE[7]	0x11C		Event received on one or more of the enabled channels in RECEIVE_CNF[n].
PUBLISH_RECEIVE[0]	0x180		Publish configuration for event RECEIVE[0]
PUBLISH_RECEIVE[1]	0x184		Publish configuration for event RECEIVE[1]
PUBLISH_RECEIVE[2]	0x188		Publish configuration for event RECEIVE[2]
PUBLISH_RECEIVE[3]	0x18C		Publish configuration for event RECEIVE[3]
PUBLISH_RECEIVE[4]	0x190		Publish configuration for event RECEIVE[4]
PUBLISH_RECEIVE[5]	0x194		Publish configuration for event RECEIVE[5]
PUBLISH_RECEIVE[6]	0x198		Publish configuration for event RECEIVE[6]
PUBLISH_RECEIVE[7]	0x19C		Publish configuration for event RECEIVE[7]
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
INTPEND	0x30C		Pending interrupts
SEND_CNF[0]	0x510		Send event configuration for TASKS_SEND[0].
SEND_CNF[1]	0x514		Send event configuration for TASKS_SEND[1].
SEND_CNF[2]	0x518		Send event configuration for TASKS_SEND[2].
SEND_CNF[3]	0x51C		Send event configuration for TASKS_SEND[3].
SEND_CNF[4]	0x520		Send event configuration for TASKS_SEND[4].
SEND_CNF[5]	0x524		Send event configuration for TASKS_SEND[5].
SEND_CNF[6]	0x528		Send event configuration for TASKS_SEND[6].
SEND_CNF[7]	0x52C		Send event configuration for TASKS_SEND[7].
RECEIVE_CNF[0]	0x590		Receive event configuration for EVENTS_RECEIVE[0].
RECEIVE_CNF[1]	0x594		Receive event configuration for EVENTS_RECEIVE[1].
RECEIVE_CNF[2]	0x598		Receive event configuration for EVENTS_RECEIVE[2].
RECEIVE_CNF[3]	0x59C		Receive event configuration for EVENTS_RECEIVE[3].
RECEIVE_CNF[4]	0x5A0		Receive event configuration for EVENTS_RECEIVE[4].
RECEIVE_CNF[5]	0x5A4		Receive event configuration for EVENTS_RECEIVE[5].
RECEIVE_CNF[6]	0x5A8		Receive event configuration for EVENTS_RECEIVE[6].
RECEIVE_CNF[7]	0x5AC		Receive event configuration for EVENTS_RECEIVE[7].
GPMEM[0]	0x610		General purpose memory.
GPMEM[1]	0x614		General purpose memory.
GPMEM[2]	0x618		General purpose memory.
GPMEM[3]	0x61C		General purpose memory.

Table 41: Register overview

6.6.1.1 TASKS_SEND[n] (n=0..7)

Address offset: 0x000 + (n × 0x4)

Trigger events on channel enabled in SEND_CNF[n].

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_SEND			Trigger events on channel enabled in SEND_CNF[n].
			Trigger	1	Trigger task

6.6.1.2 SUBSCRIBE_SEND[n] (n=0..7)

Address offset: 0x080 + (n × 0x4)

Subscribe configuration for task SEND[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task SEND[n] will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.6.1.3 EVENTS_RECEIVE[n] (n=0..7)

Address offset: 0x100 + (n × 0x4)

Event received on one or more of the enabled channels in RECEIVE_CNF[n].

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_RECEIVE			Event received on one or more of the enabled channels in RECEIVE_CNF[n].
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.6.1.4 PUBLISH_RECEIVE[n] (n=0..7)

Address offset: 0x180 + (n × 0x4)

Publish configuration for event RECEIVE[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event RECEIVE[n] will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.6.1.5 INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	RW	RECEIVE[i] (i=0..7)			Enable or disable interrupt for event RECEIVE[i]
			Disabled	0	Disable
			Enabled	1	Enable

6.6.1.6 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	RW	RECEIVE[i] (i=0..7)			Write '1' to enable interrupt for event RECEIVE[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.6.1.7 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	RW	RECEIVE[i] (i=0..7)			Write '1' to disable interrupt for event RECEIVE[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.6.1.8 INTPEND

Address offset: 0x30C

Pending interrupts

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	R	RECEIVE[i] (i=0..7)			Read pending status of interrupt for event RECEIVE[i]
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending

6.6.1.9 SEND_CNF[n] (n=0..7)

Address offset: 0x510 + (n × 0x4)

Send event configuration for TASKS_SEND[n].

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	RW	CHEN[i] (i=0..7)			Enable broadcasting on channel i.
			Disable	0	Disable broadcast.
			Enable	1	Enable broadcast.

6.6.1.10 RECEIVE_CNF[n] (n=0..7)

Address offset: 0x590 + (n × 0x4)

Receive event configuration for EVENTS_RECEIVE[n].

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A-H	RW	CHEN[i] (i=0..7)			Enable subscription to channel i.
			Disable	0	Disable events.
			Enable	1	Enable events.

6.6.1.11 GPMEM[n] (n=0..3)

Address offset: 0x610 + (n × 0x4)

General purpose memory.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	GPMEM			General purpose memory

6.7 I^{2}S — Inter-IC sound interface

The I^{2}S (Inter-IC Sound) module, supports the original two-channel I^{2}S format, and left or right-aligned formats. It implements EasyDMA for sample transfer directly to and from RAM without CPU intervention.

The I^{2}S peripheral has the following main features:

- Master and Slave mode
- Simultaneous bi-directional (TX and RX) audio streaming
- Original I^{2}S and left- or right-aligned format
- 8, 16 and 24-bit sample width
- Low-jitter Master Clock generator
- Various sample rates

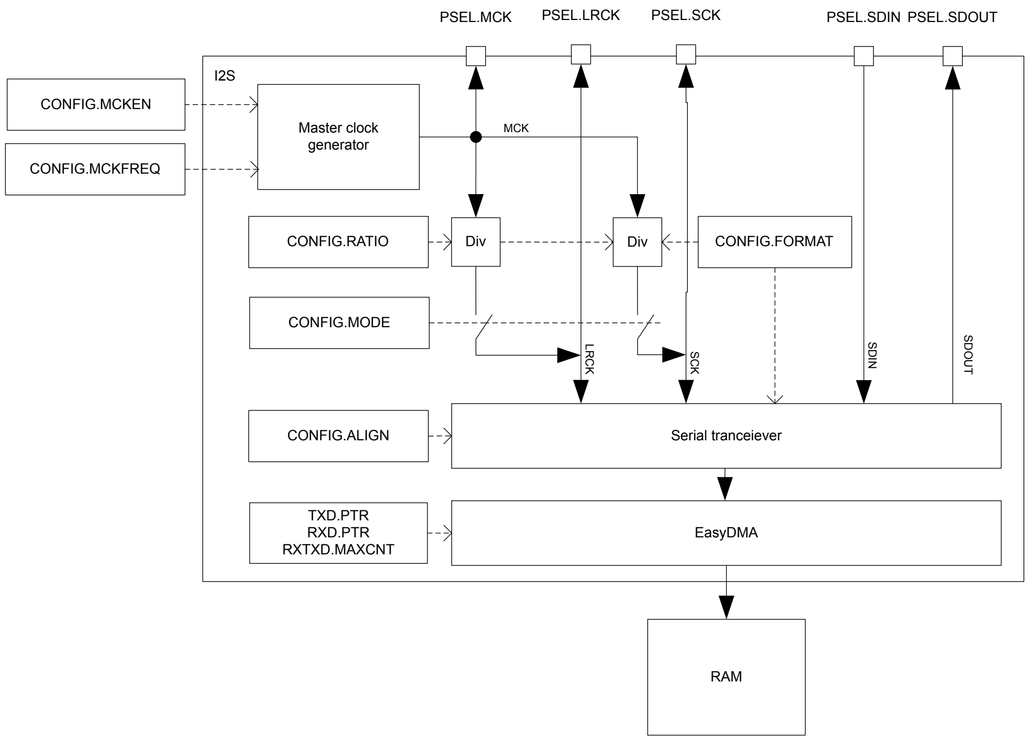


Figure 20: I^{2}S master

6.7.1 Mode

The I^{2}S protocol specification defines two modes of operation, Master and Slave.

The I^{2}S mode decides which of the two sides (Master or Slave) shall provide the clock signals LRCK and SCK, and these signals are always supplied by the Master to the Slave.

6.7.2 Transmitting and receiving

The I^{2}S module supports both transmission (TX) and reception (RX) of serial data. In both cases the serial data is shifted synchronously to the clock signals SCK and LRCK.

TX data is written to the SDOUT pin on the falling edge of SCK, and RX data is read from the SDIN pin on the rising edge of SCK. The most significant bit (MSB) is always transmitted first.

TX and RX are available in both Master and Slave modes and can be enabled/disabled independently in the CONFIG.TXEN on page 130 and CONFIG.RXEN on page 129.

Transmission and/or reception is started by triggering the START task. When started and transmission is enabled (in CONFIG.TXEN on page 130), the TXPTRUPD event will be generated for every RXTXD.MAXCNT on page 133 number of transmitted data words (containing one or more samples). Similarly, when started and reception is enabled (in CONFIG.RXEN on page 129), the RXPTRUPD event will be generated for every RXTXD.MAXCNT on page 133 received data words.

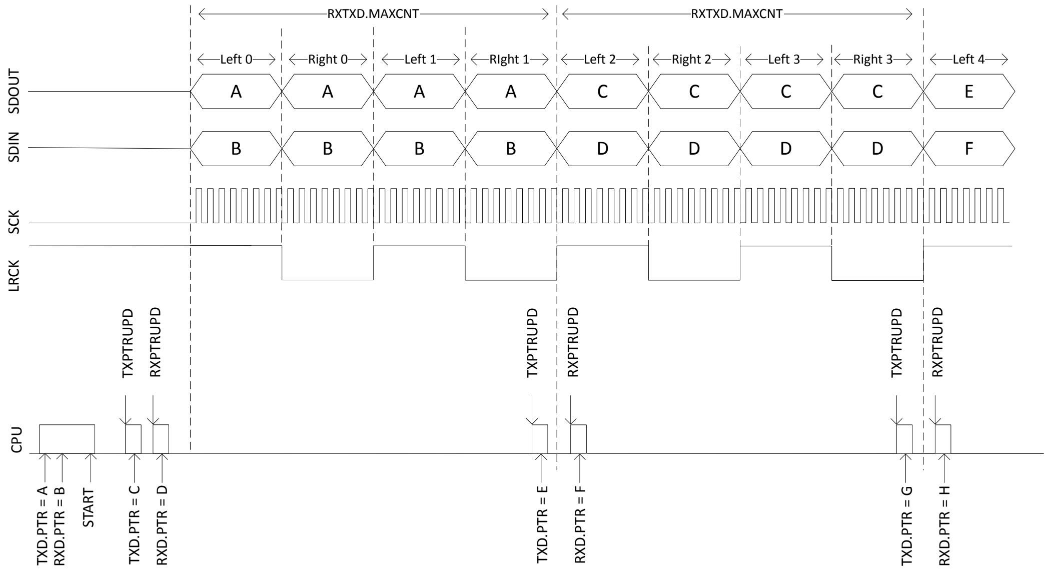


Figure 21: Transmitting and receiving. CONFIG.FORMAT = Aligned, CONFIG.SWIDTH = 8Bit, CONFIG.CHANNELS = Stereo, RXTXD.MAXCNT = 1.

6.7.3 Left right clock (LRCK)

The Left Right Clock (LRCK), often referred to as "word clock", "sample clock" or "word select" in I^{2}S context, is the clock defining the frames in the serial bit streams sent and received on SDOUT and SDIN, respectively.

In I2S mode, each frame contains one left and right sample pair, with the left sample being transferred during the low half period of LRCK followed by the right sample being transferred during the high period of LRCK.

In Aligned mode, each frame contains one left and right sample pair, with the left sample being transferred during the high half period of LRCK followed by the right sample being transferred during the low period of LRCK.

Consequently, the LRCK frequency is equivalent to the audio sample rate.

When operating in Master mode, the LRCK is generated from the MCK, and the frequency of LRCK is then given as:

```
LRCK = MCK / CONFIG.RATIO
```

LRCK always toggles around the falling edge of the serial clock SCK.

6.7.4 Serial clock (SCK)

The serial clock (SCK), often referred to as the serial bit clock, pulses once for each data bit being transferred on the serial data lines SDIN and SDOUT.

When operating in Master mode the SCK is generated from the MCK, and the frequency of SCK is then given as:

```
SCK = 2 * LRCK * CONFIG.SWIDTH
```

The falling edge of the SCK falls on the toggling edge of LRCK.

When operating in Slave mode SCK is provided by the external I^{2}S master.

6.7.5 Master clock (MCK)

The master clock (MCK) is the clock from which LRCK and SCK are derived when operating in Master mode.

The MCK is generated by an internal MCK generator. This generator always needs to be enabled when in Master mode, but the generator can also be enabled when in Slave mode. Enabling the generator when in slave mode can be useful in the case where the external Master is not able to generate its own master clock.

The MCK generator is enabled/disabled in the register CONFIG.MCKEN on page 130, and the generator is started or stopped by the START or STOP tasks.

In Master mode the LRCK and the SCK frequencies are closely related, as both are derived from MCK and set indirectly through CONFIG.RATIO on page 131 and CONFIG.SWIDTH on page 131.

When configuring these registers, the user is responsible for fulfilling the following requirements:

1. SCK frequency can never exceed the MCK frequency, which can be formulated as:

   ```
   CONFIG.RATIO >= 2 * CONFIG.SWIDTH
   ```

2. The MCK/LRCK ratio shall be a multiple of 2 * CONFIG.SWIDTH, which can be formulated as:

```
Integer = (CONFIG.RATIO / (2 * CONFIG.SWIDTH))
```

The MCK signal can be routed to an output pin (specified in PSEL.MCK) to supply external I^{2}S devices that require the MCK to be supplied from the outside.

When operating in Slave mode, the I^{2}S module does not use the MCK and the MCK generator does not need to be enabled.

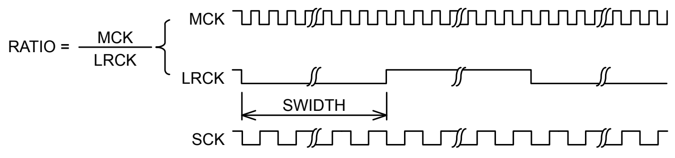


Figure 22: Relation between RATIO, MCK and LRCK.

Desired LRCK [Hz]	CONFIG.SWIDTH	CONFIG.RATIO	CONFIG.MCKFREQ	MCK [Hz]	LRCK [Hz]	LRCK error [%]
16000	16Bit	32X	32MDIV63	507936.5	15873.0	-0.8
16000	16Bit	64X	32MDIV31	1032258.1	16129.0	0.8
16000	16Bit	256X	32MDIV8	4000000.0	15625.0	-2.3
32000	16Bit	32X	32MDIV31	1032258.1	32258.1	0.8
32000	16Bit	64X	32MDIV16	2000000.0	31250.0	-2.3
44100	16Bit	32X	32MDIV23	1391304.3	43478.3	-1.4
44100	16Bit	64X	32MDIV11	2909090.9	45454.5	3.1

Table 42: Configuration examples

6.7.6 Width, alignment and format

The CONFIG.SWIDTH register primarily defines the sample width of the data written to memory. In master mode, it then also sets the amount of bits per frame. In Slave mode it controls padding/trimming if required. Left, right, transmitted, and received samples always have the same width. The CONFIG.FORMAT register specifies the position of the data frames with respect to the LRCK edges in both Master and Slave modes.

When using I^{2}S format, the first bit in a half-frame (containing one left or right sample) gets sampled on the second rising edge of the SCK after a LRCK edge. When using Aligned mode, the first bit in a half-frame gets sampled on the first rising edge of SCK following a LRCK edge.

For data being received on SDIN the sample value can be either right or left-aligned inside a half-frame, as specified in CONFIG.ALIGN on page 131. CONFIG.ALIGN on page 131 affects only the decoding of the incoming samples (SDIN), while the outgoing samples (SDOUT) are always left-aligned (or justified).

When using left-alignment, each half-frame starts with the MSB of the sample value (both for data being sent on SDOUT and received on SDIN).

When using right-alignment, each half-frame of data being received on SDIN ends with the LSB of the sample value, while each half-frame of data being sent on SDOUT starts with the MSB of the sample value (same as for left-alignment).

In Master mode, the size of a half-frame (in number of SCK periods) equals the sample width (in number of bits), and in this case the alignment setting does not care as each half-frame in any case will start with the MSB and end with the LSB of the sample value.

In slave mode, however, the sample width does not need to equal the frame size. This means you might have extra or fewer SCK pulses per half-frame than what the sample width specified in CONFIG.SWIDTH requires.

In the case where we use **left-alignment** and the number of SCK pulses per half-frame is **higher** than the sample width, the following will apply:

- For data received on SDIN, all bits after the LSB of the sample value will be discarded.
- For data sent on SDOUT, all bits after the LSB of the sample value will be 0.

In the case where we use **left-alignment** and the number of SCK pulses per frame is **lower** than the sample width, the following will apply:

- Data sent and received on SDOUT and SDIN will be truncated with the LSBs being removed first.

In the case where we use **right-alignment** and the number of SCK pulses per frame is **higher** than the sample width, the following will apply:

- For data received on SDIN, all bits before the MSB of the sample value will be discarded.
- For data sent on SDOUT, all bits after the LSB of the sample value will be 0 (same behavior as for left-alignment).

In the case where we use **right-alignment** and the number of SCK pulses per frame is **lower** than the sample width, the following will apply:

- Data received on SDIN will be sign-extended to "sample width" number of bits before being written to memory.
- Data sent on SDOUT will be truncated with the LSBs being removed first (same behavior as for left-alignment).

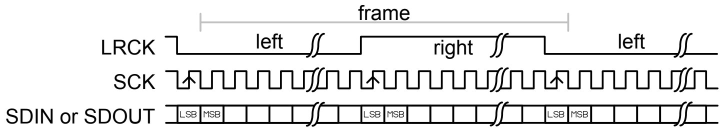


Figure 23: I^{2}S format. CONFIG.SWIDTH equalling half-frame size.

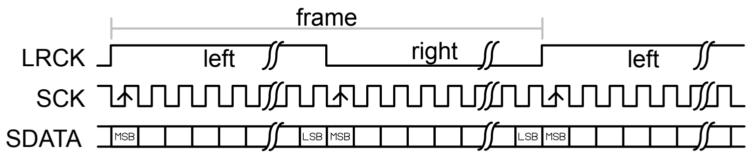


Figure 24: Aligned format. CONFIG.SWIDTH equalling half-frame size.

6.7.7 EasyDMA

The I^{2}S module implements EasyDMA for accessing internal Data RAM without CPU intervention.

The source and destination pointers for the TX and RX data are configured in TXD.PTR on page 132 and RXD.PTR on page 132. The memory pointed to by these pointers will only be read or written when TX or RX are enabled in CONFIG.TXEN on page 130 and CONFIG.RXEN on page 129.

The addresses written to the pointer registers TXD.PTR on page 132 and RXD.PTR on page 132 are double-buffered in hardware, and these double buffers are updated for every RXTXD.MAXCNT on page 133 words (containing one or more samples) read/written from/to memory. The events TXPTRUPD and RXPTRUPD are generated whenever the TXD.PTR and RXD.PTR are transferred to these double buffers.

If TXD.PTR on page 132 is not pointing to the Data RAM region when transmission is enabled, or RXD.PTR on page 132 is not pointing to the Data RAM region when reception is enabled, an EasyDMA transfer may result in a HardFault and/or memory corruption. See Memory on page 20 for more information about the different memory regions.

Due to the nature of I^{2}S, where the number of transmitted samples always equals the number of received samples (at least when both TX and RX are enabled), one common register RXTXD.MAXCNT on page 133 is used for specifying the sizes of these two memory buffers. The size of the buffers is specified in a number of 32-bit words. Such a 32-bit memory word can either contain four 8-bit samples, two 16-bit samples or one right-aligned 24-bit sample sign extended to 32 bit.

In stereo mode (CONFIG.CHANNELS=Stereo), the samples are stored as "left and right sample pairs" in memory. Figure Memory mapping for 8 bit stereo. CONFIG.SWIDTH = 8Bit, CONFIG.CHANNELS = Stereo. on page 120, Memory mapping for 16 bit stereo. CONFIG.SWIDTH = 16Bit, CONFIG.CHANNELS = Stereo. on page 120 and Memory mapping for 24 bit stereo. CONFIG.SWIDTH = 24Bit, CONFIG.CHANNELS = Stereo. on page 121 show how the samples are mapped to memory in this mode. The mapping is valid for both RX and TX.

In mono mode (CONFIG.CHANNELS=Left or Right), RX sample from only one channel in the frame is stored in memory, the other channel sample is ignored. Illustrations Memory mapping for 8 bit mono. CONFIG.SWIDTH = 8Bit, CONFIG.CHANNELS = Left. on page 120, Memory mapping for 16 bit mono, left channel only. CONFIG.SWIDTH = 16Bit, CONFIG.CHANNELS = Left. on page 120 and Memory mapping for 24 bit mono, left channel only. CONFIG.SWIDTH = 24Bit, CONFIG.CHANNELS = Left. on page 121 show how RX samples are mapped to memory in this mode.

For TX, the same outgoing sample read from memory is transmitted on both left and right in a frame, resulting in a mono output stream.

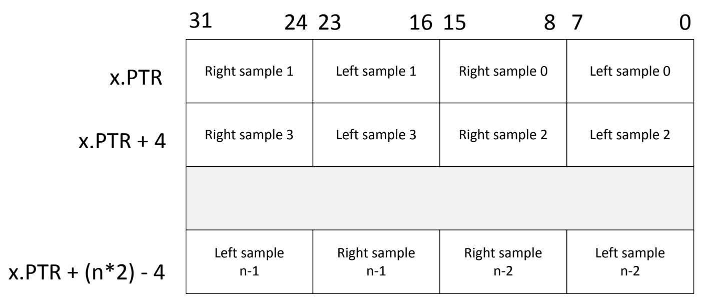


Figure 25: Memory mapping for 8 bit stereo. CONFIG.SWIDTH = 8Bit, CONFIG.CHANNELS = Stereo.

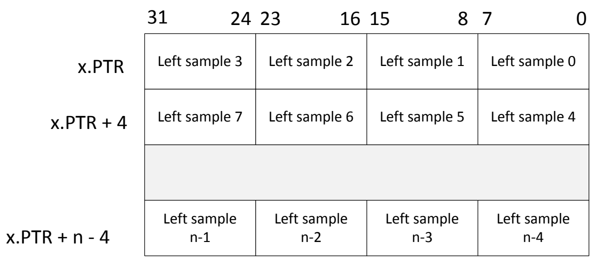


Figure 26: Memory mapping for 8 bit mono. CONFIG.SWIDTH = 8Bit, CONFIG.CHANNELS = Left.

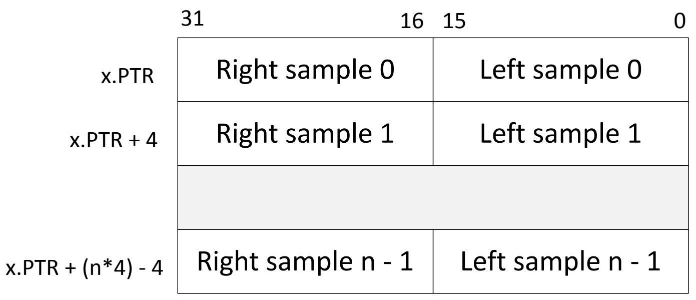


Figure 27: Memory mapping for 16 bit stereo. CONFIG.SWIDTH = 16Bit, CONFIG.CHANNELS = Stereo.

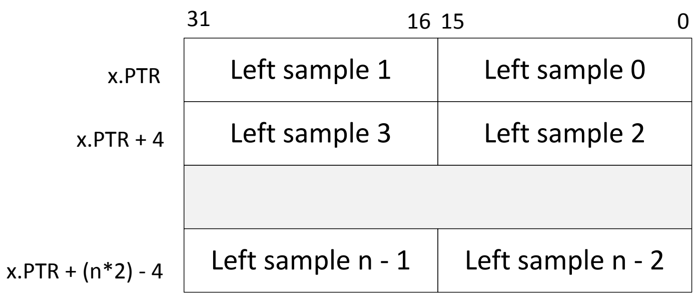


Figure 28: Memory mapping for 16 bit mono, left channel only. CONFIG.SWIDTH = 16Bit, CONFIG.CHANNELS = Left.

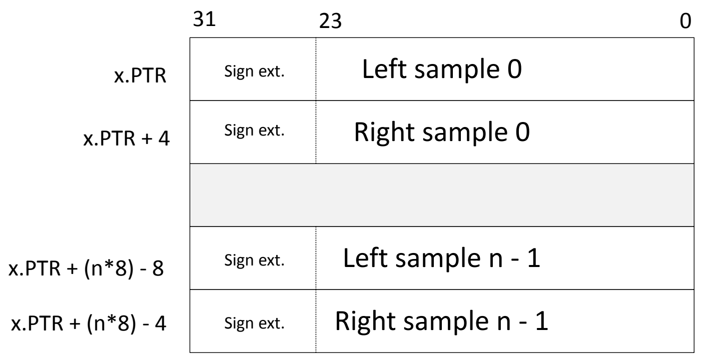


Figure 29: Memory mapping for 24 bit stereo. CONFIG.SWIDTH = 24Bit, CONFIG.CHANNELS = Stereo.

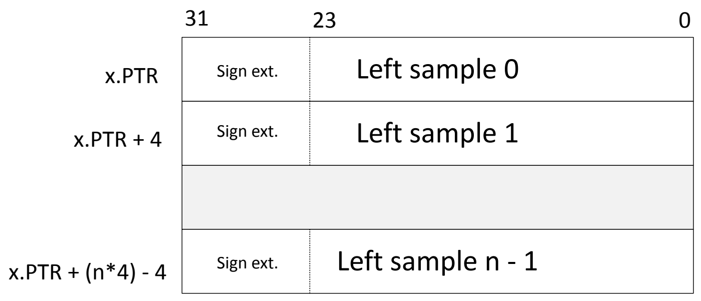


Figure 30: Memory mapping for 24 bit mono, left channel only. CONFIG.SWIDTH = 24Bit, CONFIG.CHANNELS = Left.

6.7.8 Module operation

Described here is a typical operating procedure for the I^{2}S module.

1. Configure the I^{2}S module using the CONFIG registers

```
// Enable reception
NRF_I2S->CONFIG.RXEN = (I2S_CONFIG_RXEN_RXEN_Enabled <<
                                       I2S_CONFIG_RXEN_RXEN_Pos);
// Enable transmission
NRF_I2S->CONFIG.TXEN = (I2S_CONFIG_TXEN_TXEN_Enabled <<
                                       I2S_CONFIG_TXEN_TXEN_Pos);
// Enable MCK generator
NRF_I2S->CONFIG.MCKEN = (I2S_CONFIG_MCKEN_MCKEN_Enabled <<
                                       I2S_CONFIG_MCKEN_MCKEN_Pos);
// MCKFREQ = 4 MHz
NRF_I2S->CONFIG.MCKFREQ = I2S_CONFIG_MCKFREQ_MCKFREQ_32MDIV8 <<
                                       I2S_CONFIG_MCKFREQ_MCKFREQ_Pos;
// Ratio = 256
NRF_I2S->CONFIG.RATIO = I2S_CONFIG_RATIO_RATIO_256X <<
                                       I2S_CONFIG_RATIO_RATIO_Pos;
// MCKFREQ = 4 MHz and Ratio = 256 gives sample rate = 15.625 ks/s
// Sample width = 16 bit
NRF_I2S->CONFIG.SWIDTH = I2S_CONFIG_SWIDTH_SWIDTH_16Bit <<
                                       I2S_CONFIG_SWIDTH_SWIDTH_Pos;
// Alignment = Left
NRF_I2S->CONFIG.ALIGN = I2S_CONFIG_ALIGN_ALIGN_Left <<
                                       I2S_CONFIG_ALIGN_ALIGN_Pos;
// Format = I2S
NRF_I2S->CONFIG.FORMAT = I2S_CONFIG_FORMAT_FORMAT_I2S <<
                                       I2S_CONFIG_FORMAT_FORMAT_Pos;
// Use stereo
NRF_I2S->CONFIG.CHANNELS = I2S_CONFIG_CHANNELS_CHANNELS_Stereo <<
                                       I2S_CONFIG_CHANNELS_CHANNELS_Pos;
```

2. Map IO pins using the PINSEL registers

```
// MCK routed to pin 0
NRF_I2S->PSEL.MCK = (0 << I2S_PSEL_MCK_PIN_Pos) |
                    (I2S_PSEL_MCK_CONNECT_Connected <<
                                                 I2S_PSEL_MCK_CONNECT_Pos);
// SCK routed to pin 1
NRF_I2S->PSEL.SCK = (1 << I2S_PSEL_SCK_PIN_Pos) |
                    (I2S_PSEL_SCK_CONNECT_Connected <<
                                                 I2S_PSEL_SCK_CONNECT_Pos);
// LRCK routed to pin 2
NRF_I2S->PSEL.LRCK = (2 << I2S_PSEL_LRCK_PIN_Pos) |
                     (I2S_PSEL_LRCK_CONNECT_Connected <<
                                                 I2S_PSEL_LRCK_CONNECT_Pos);
// SDOUT routed to pin 3
NRF_I2S->PSEL.SDOUT = (3 << I2S_PSEL_SDOUT_PIN_Pos) |
                      (I2S_PSEL_SDOUT_CONNECT_Connected <<
                                                 I2S_PSEL_SDOUT_CONNECT_Pos);
// SDIN routed on pin 4
NRF_I2S->PSEL.SDIN = (4 << I2S_PSEL_SDIN_PIN_Pos) |
                     (I2S_PSEL_SDIN_CONNECT_Connected <<
                                                 I2S_PSEL_SDIN_CONNECT_Pos);
```

3. Configure TX and RX data pointers using the TXD, RXD and RXTXD registers

```
NRF_I2S->TXD.PTR = my_tx_buf;
NRF_I2S->RXD.PTR = my_rx_buf;
NRF_I2S->TXD.MAXCNT = MY_BUF_SIZE;
```

4. Enable the I^{2}S module using the ENABLE register

```
NRF_I2S->ENABLE = 1;
```

5. Start audio streaming using the START task

```
NRF_I2S->TASKS_START = 1;
```

6. Handle received and transmitted data when receiving the TXPTRUPD and RXPTRUPD events

```
if(NRF_I2S->EVENTS_TXPTRUPD  != 0)
{
    NRF_I2S->TXD.PTR = my_next_tx_buf;
    NRF_I2S->EVENTS_TXPTRUPD = 0;
}

if(NRF_I2S->EVENTS_RXPTRUPD != 0)
{
    NRF_I2S->RXD.PTR = my_next_rx_buf;
    NRF_I2S->EVENTS_RXPTRUPD = 0;
}
```

6.7.9 Pin configuration

The MCK, SCK, LRCK, SDIN and SDOUT signals associated with the I^{2}S module are mapped to physical pins according to the pin numbers specified in the PSEL.x registers.

These pins are acquired whenever the I^{2}S module is enabled through the register ENABLE on page 129.

When a pin is acquired by the I^{2}S module, the direction of the pin (input or output) will be configured automatically, and any pin direction setting done in the GPIO module will be overridden. The directions for the various I^{2}S pins are shown below in GPIO configuration before enabling peripheral (master mode) on page 123 and GPIO configuration before enabling peripheral (slave mode) on page 124.

To secure correct signal levels on the pins when the system is in OFF mode, and when the I^{2}S module is disabled, these pins must be configured in the GPIO peripheral directly.

I^{2}S signal	I^{2}S pin	Direction	Output value	Comment
MCK	As specified in PSEL.MCK	Output	0	
LRCK	As specified in PSEL.LRCK	Output	0	
SCK	As specified in PSEL.SCK	Output	0	
SDIN	As specified in PSEL.SDIN	Input	Not applicable	
SDOUT	As specified in PSEL.SDOUT	Output	0	

Table 43: GPIO configuration before enabling peripheral (master mode)

I^2S signal	I^2S pin	Direction	Output value	Comment
MCK	As specified in PSEL.MCK	Output	0	
LRCK	As specified in PSEL.LRCK	Input	Not applicable	
SCK	As specified in PSEL.SCK	Input	Not applicable	
SDIN	As specified in PSEL.SDIN	Input	Not applicable	
SDOUT	As specified in PSEL.SDOUT	Output	0	

Table 44: GPIO configuration before enabling peripheral (slave mode)

6.7.10 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50028000 0x40028000	I2S	I2S : S I2S : NS	US	SA	Inter-IC Sound	

Table 45: Instances

Register	Offset	Security	Description
TASKS_START	0x000		Starts continuous I2S transfer. Also starts MCK generator when this is enabled.
TASKS_STOP	0x004		Stops I2S transfer. Also stops MCK generator. Triggering this task will cause the STOPPED event to be generated.
SUBSCRIBE_START	0x080		Subscribe configuration for task START
SUBSCRIBE_STOP	0x084		Subscribe configuration for task STOP
EVENTS_RXPTRUPD	0x104		The RXD.PTR register has been copied to internal double-buffers. When the I2S module is started and RX is enabled, this event will be generated for every RXTXD.MAXCNT words that are received on the SDIN pin.
EVENTS_STOPPED	0x108		I2S transfer stopped.
EVENTS_TXPTRUPD	0x114		The TDX.PTR register has been copied to internal double-buffers. When the I2S module is started and TX is enabled, this event will be generated for every RXTXD.MAXCNT words that are sent on the SDOUT pin.
PUBLISH_RXPTRUPD	0x184		Publish configuration for event RXPTRUPD
PUBLISH_STOPPED	0x188		Publish configuration for event STOPPED
PUBLISH_TXPTRUPD	0x194		Publish configuration for event TXPTRUPD
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
ENABLE	0x500		Enable I2S module.
CONFIG.MODE	0x504		I2S mode.
CONFIG.RXEN	0x508		Reception (RX) enable.
CONFIG.TXEN	0x50C		Transmission (TX) enable.
CONFIG.MCKEN	0x510		Master clock generator enable.
CONFIG.MCKFREQ	0x514		Master clock generator frequency.
CONFIG.RATIO	0x518		MCK / LRCK ratio.
CONFIG.SWIDTH	0x51C		Sample width.
CONFIG.ALIGN	0x520		Alignment of sample within a frame.
CONFIG.FORMAT	0x524		Frame format.
CONFIG.CHANNELS	0x528		Enable channels.
RXD.PTR	0x538		Receive buffer RAM start address.
TXD.PTR	0x540		Transmit buffer RAM start address.
RXTXD.MAXCNT	0x550		Size of RXD and TXD buffers.
PSEL.MCK	0x560		Pin select for MCK signal.
PSEL.SCK	0x564		Pin select for SCK signal.
PSEL.LRCK	0x568		Pin select for LRCK signal.

Register	Offset	Security	Description
PSEL.SDIN	0x56C		Pin select for SDIN signal.
PSEL.SDOUT	0x570		Pin select for SDOUT signal.

Table 46: Register overview

6.7.10.1 TASKS_START

Address offset: 0x000

Starts continuous I2S transfer. Also starts MCK generator when this is enabled.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_START			Starts continuous I2S transfer. Also starts MCK generator when this is enabled.
			Trigger	1	Trigger task

6.7.10.2 TASKS_STOP

Address offset: 0x004

Stops I2S transfer. Also stops MCK generator. Triggering this task will cause the STOPPED event to be generated.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_STOP			Stops I2S transfer. Also stops MCK generator. Triggering this task will cause the STOPPED event to be generated.
			Trigger	1	Trigger task

6.7.10.3 SUBSCRIBE_START

Address offset: 0x080

Subscribe configuration for task START

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task START will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.7.10.4 SUBSCRIBE_STOP

Address offset: 0x084

Subscribe configuration for task STOP

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task STOP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.7.10.5 EVENTS_RXPTRUPD

Address offset: 0x104

The RXD.PTR register has been copied to internal double-buffers. When the I2S module is started and RX is enabled, this event will be generated for every RXTXD.MAXCNT words that are received on the SDIN pin.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_RXPTRUPD			The RXD.PTR register has been copied to internal double-buffers. When the I2S module is started and RX is enabled, this event will be generated for every RXTXD.MAXCNT words that are received on the SDIN pin.
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.7.10.6 EVENTS_STOPPED

Address offset: 0x108

I2S transfer stopped.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_STOPPED			I2S transfer stopped.
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.7.10.7 EVENTS_TXPTRUPD

Address offset: 0x114

The TDX.PTR register has been copied to internal double-buffers. When the I2S module is started and TX is enabled, this event will be generated for every RXTXD.MAXCNT words that are sent on the SDOUT pin.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_TXPTRUPD			The TDX.PTR register has been copied to internal double-buffers. When the I2S module is started and TX is enabled, this event will be generated for every RXTXD.MAXCNT words that are sent on the SDOUT pin.
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.7.10.8 PUBLISH_RXPTRUPD

Address offset: 0x184

Publish configuration for event RXPTRUPD

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event RXPTRUPD will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.7.10.9 PUBLISH_STOPPED

Address offset: 0x188

Publish configuration for event STOPPED

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event STOPPED will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.7.10.10 PUBLISH_TXPTRUPD

Address offset: 0x194

Publish configuration for event TXPTRUPD

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event TXPTRUPD will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.7.10.11 INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																											F			C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
B	RW	RXPTRUPD			Enable or disable interrupt for event RXPTRUPD
			Disabled	0	Disable
			Enabled	1	Enable
C	RW	STOPPED			Enable or disable interrupt for event STOPPED
			Disabled	0	Disable
			Enabled	1	Enable
F	RW	TXPTRUPD			Enable or disable interrupt for event TXPTRUPD
			Disabled	0	Disable
			Enabled	1	Enable

6.7.10.12 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																											F			C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
B	RW	RXPTRUPD			Write '1' to enable interrupt for event RXPTRUPD
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	STOPPED			Write '1' to enable interrupt for event STOPPED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
F	RW	TXPTRUPD			Write '1' to enable interrupt for event TXPTRUPD
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.7.10.13 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																											F			C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
B	RW	RXPTRUPD			Write '1' to disable interrupt for event RXPTRUPD
			Clear	1	Disable

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																											F			C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	STOPPED			Write '1' to disable interrupt for event STOPPED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
F	RW	TXPTRUPD			Write '1' to disable interrupt for event TXPTRUPD
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.7.10.14 ENABLE

Address offset: 0x500

Enable I2S module.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	ENABLE			Enable I2S module.
			Disabled	0	Disable
			Enabled	1	Enable

6.7.10.15 CONFIG.MODE

Address offset: 0x504

I2S mode.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	MODE			I2S mode.
			Master	0	Master mode. SCK and LRCK generated from internal master clcok (MCK) and output on pins defined by PSEL.xxx.
			Slave	1	Slave mode. SCK and LRCK generated by external master and received on pins defined by PSEL.xxx

6.7.10.16 CONFIG.RXEN

Address offset: 0x508

Reception (RX) enable.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	RXEN			Reception (RX) enable.
			Disabled	0	Reception disabled and now data will be written to the RXD.PTR address.
			Enabled	1	Reception enabled.

6.7.10.17 CONFIG.TXEN

Address offset: 0x50C

Transmission (TX) enable.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000001	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	TXEN			Transmission (TX) enable.
			Disabled	0	Transmission disabled and now data will be read from the RXD.TXD address.
			Enabled	1	Transmission enabled.

6.7.10.18 CONFIG.MCKEN

Address offset: 0x510

Master clock generator enable.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000001	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	MCKEN			Master clock generator enable.
			Disabled	0	Master clock generator disabled and PSEL.MCK not connected(available as GPIO).
			Enabled	1	Master clock generator running and MCK output on PSEL.MCK.

6.7.10.19 CONFIG.MCKFREQ

Address offset: 0x514

Master clock generator frequency.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x20000000	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	MCKFREQ			Master clock generator frequency.
			32MDIV8	0x20000000	32 MHz / 8 = 4.0 MHz
			32MDIV10	0x18000000	32 MHz / 10 = 3.2 MHz
			32MDIV11	0x16000000	32 MHz / 11 = 2.9090909 MHz
			32MDIV15	0x11000000	32 MHz / 15 = 2.1333333 MHz

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x20000000	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
			32MDIV16	0x10000000	32 MHz / 16 = 2.0 MHz
			32MDIV21	0x0C000000	32 MHz / 21 = 1.5238095
			32MDIV23	0x0B000000	32 MHz / 23 = 1.3913043 MHz
			32MDIV30	0x08800000	32 MHz / 30 = 1.0666667 MHz
			32MDIV31	0x08400000	32 MHz / 31 = 1.0322581 MHz
			32MDIV32	0x08000000	32 MHz / 32 = 1.0 MHz
			32MDIV42	0x06000000	32 MHz / 42 = 0.7619048 MHz
			32MDIV63	0x04100000	32 MHz / 63 = 0.5079365 MHz
			32MDIV125	0x020C0000	32 MHz / 125 = 0.256 MHz

6.7.10.20 CONFIG.RATIO

Address offset: 0x518

MCK / LRCK ratio.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																													A	A	A	A
Reset 0x00000006	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	RATIO			MCK / LRCK ratio.
			32X	0	LRCK = MCK / 32
			48X	1	LRCK = MCK / 48
			64X	2	LRCK = MCK / 64
			96X	3	LRCK = MCK / 96
			128X	4	LRCK = MCK / 128
			192X	5	LRCK = MCK / 192
			256X	6	LRCK = MCK / 256
			384X	7	LRCK = MCK / 384
			512X	8	LRCK = MCK / 512

6.7.10.21 CONFIG.SWIDTH

Address offset: 0x51C

Sample width.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															A	A
Reset 0x00000001	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	SWIDTH			Sample width.
			8Bit	0	8 bit.
			16Bit	1	16 bit.
			24Bit	2	24 bit.

6.7.10.22 CONFIG.ALIGN

Address offset: 0x520

Alignment of sample within a frame.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	ALIGN			Alignment of sample within a frame.
			Left	0	Left-aligned.
			Right	1	Right-aligned.

6.7.10.23 CONFIG.FORMAT

Address offset: 0x524

Frame format.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	FORMAT			Frame format.
			I2S	0	Original I2S format.
			Aligned	1	Alternate (left- or right-aligned) format.

6.7.10.24 CONFIG.CHANNELS

Address offset: 0x528

Enable channels.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHANNELS			Enable channels.
			Stereo	0	Stereo.
			Left	1	Left only.
			Right	2	Right only.

6.7.10.25 RXD.PTR

Address offset: 0x538

Receive buffer RAM start address.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	PTR			Receive buffer Data RAM start address. When receiving, words containing samples will be written to this address. This address is a word aligned Data RAM address.

6.7.10.26 TXD.PTR

Address offset: 0x540

Transmit buffer RAM start address.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	PTR			Transmit buffer Data RAM start address. When transmitting, words containing samples will be fetched from this address. This address is a word aligned Data RAM address.

6.7.10.27 RXTXD.MAXCNT

Address offset: 0x550

Size of RXD and TXD buffers.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																			A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	MAXCNT			Size of RXD and TXD buffers in number of 32 bit words.

6.7.10.28 PSEL.MCK

Address offset: 0x560

Pin select for MCK signal.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C																											A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

6.7.10.29 PSEL.SCK

Address offset: 0x564

Pin select for SCK signal.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C																											A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

6.7.10.30 PSEL.LRCK

Address offset: 0x568

Pin select for LRCK signal.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C																											A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

6.7.10.31 PSEL.SDIN

Address offset: 0x56C

Pin select for SDIN signal.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C																											A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

6.7.10.32 PSEL.SDOUT

Address offset: 0x570

Pin select for SDOUT signal.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C																											A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

6.7.11 Electrical specification

6.7.11.1 I2S timing specification

Symbol	**Description**	**Min.**	**Typ.**	**Max.**	**Units**
t_{S_SDIN}	SDIN setup time before SCK rising	20			ns
t_{H_SDIN}	SDIN hold time after SCK rising	15			ns
t_{S_SDOUT}	SDOUT setup time after SCK falling	40			ns
t_{H_SDOUT}	SDOUT hold time before SCK falling	6			ns
t_{SCK_LRCK}	SCLK falling to LRCK edge	-5	0	5	ns
f_{MCK}	MCK frequency			4000	kHz
f_{LRCK}	LRCK frequency			48	kHz
f_{SCK}	SCK frequency			2000	kHz
DC_{CK}	Clock duty cycle (MCK, LRCK, SCK)	45		55	%

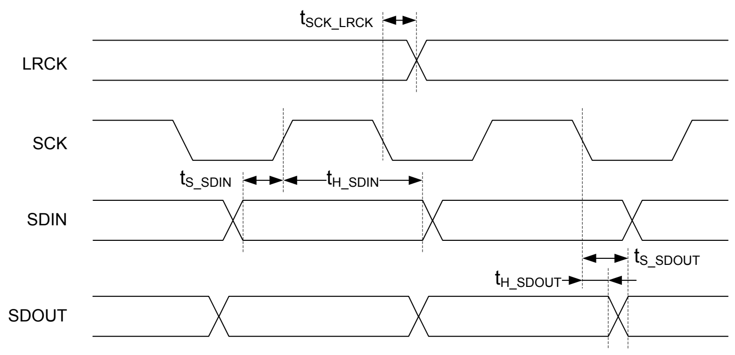


Figure 31: I2S timing diagram

6.8 KMU — Key management unit

Key management unit (KMU) is a component of the NVMC for secure key handling. KMU uses a subset of the flash referred to as user information configuration register (UICR) for its secure storage. This UICR subset can be used for both establishing a device root of trust (RoT) during chip and OEM manufacturing, and for storage and use of any device specific keys.

Access and use of information stored in UICR is controlled through the KMU. Even though the KMU and UICR are tightly coupled, they do not share a common memory map:

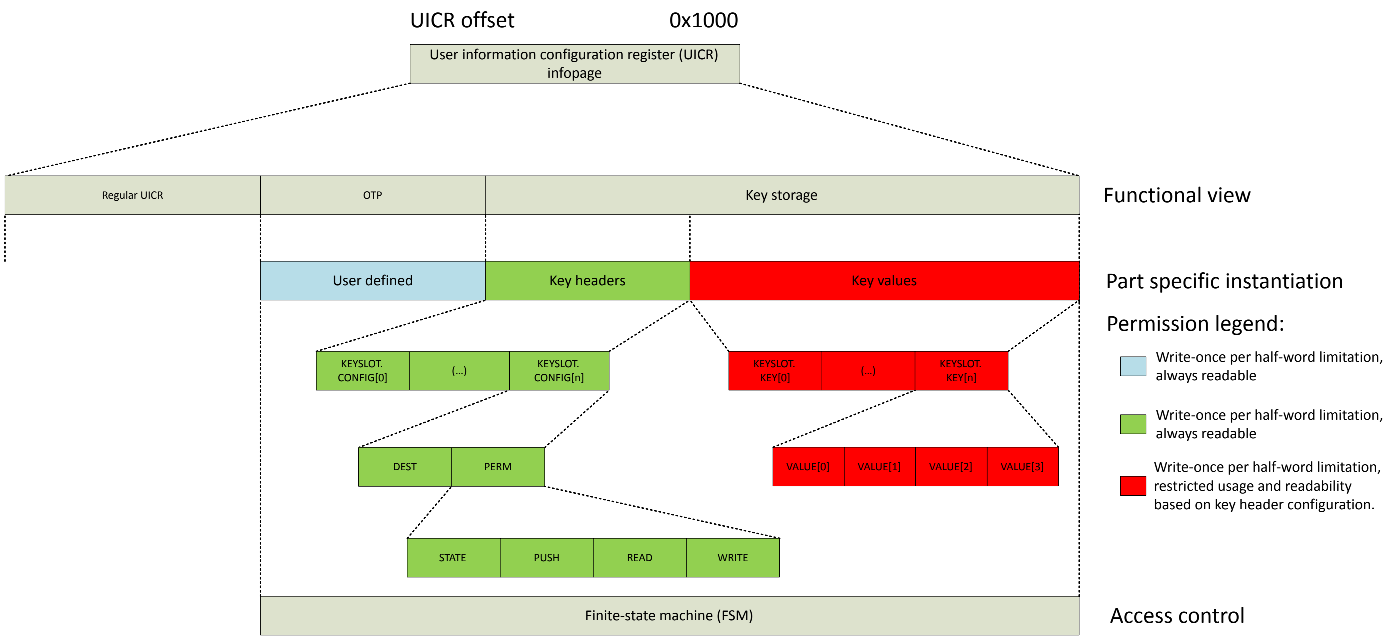

UICR offset
0x1000
User information configuration register (UICR) infopage
Regular UICR
OTP
Key storage
Functional view
User defined
Key headers
Key values
Part specific instantiation
Permission legend:
Write-once per half-word limitation, always readable
Write-once per half-word limitation, always readable
Write-once per half-word limitation, restricted usage and readability based on key header configuration.
KEYSLOT. CONFIG[0]
(...)
KEYSLOT. CONFIG[n]
KEYSLOT. KEY[0]
(...)
KEYSLOT. KEY[n]
DEST
PERM
VALUE[0]
VALUE[1]
VALUE[2]
VALUE[3]
STATE
PUSH
READ
WRITE
Finite-state machine (FSM)
Access control

Figure 32: Memory map overview

The KMU is mapped as a stand-alone peripheral on the APB bus, while UICR is addressable on AHB and is located in flash memory map. Access to the KMU and keys stored in UICR is only allowed in secure mode. Access to the UICR memory map is equivalent to any other flash page access, except that the KMU will enforce usage and read/write restrictions to different regions of the UICR memory map depending on configuration.

For more information about the user information configuration registers, see chapter UICR — User information configuration registers on page 41.

6.8.1 Functional view

From a functional view UICR is divided into two different regions:

- One-time programmable (OTP) memory
- Key storage

OTP

One-time programmable (OTP) memory is typically used for holding values that are written once, and then never to be changed throughout the life-time of the product. The OTP region of UICR is emulated by placing a write-once per halfword limitation on registers defined here.

Key storage

The key storage region contains multiple key slots, where each slot consists of a key header and an associated key value. The key value is limited to 128 bits. Any key size greater than 128 bits must be divided and distributed over multiple key slot instances.

Key headers are allocated an address range of 0x400 in the UICR memory map, allowing for a total of 128 keys to be addressable inside the key storage region.

> **Note:** Use of the key storage region in UICR should be limited to keys with a certain life-span, and not per-session derived keys where the CPU is involved in the key exchange.

6.8.2 Access control

Access control to the underlying UICR infopage in the flash is enforced by a hardware finite-state machine (FSM). FSM can allow or block transactions depending both on the security of the transaction (secure or non-secure) and the type of register being written and/or read.

Access type	Key headers	Key values
Read	Allowed	Restricted
Write	Restricted	Restricted

Table 47: Access control

Any restricted access requires an explicit key slot selection through the KMU register interface. Any illegal access to restricted key slot registers will be blocked and word `0xDEADDEAD` will be returned on AHB.

The OTP region has individual access control behavior, while access control to the key storage region is configured on a per key slot basis. KMU FSM operates on only one key slot instance at a time, and the permissions and usage restriction for a key value associated with a key slot can be configured individually.

> **Note:** Even if the KMU can be configured as non-secure, all non-secure transactions will be blocked.

6.8.3 Protecting UICR content

UICR content can be protected against device-internal `NVMC->ERASEALL` requests, in addition to device-external `ERASEALL` requests, through the CTRL-AP interface. This feature is useful if the firmware designers want to prevent the OTP region from being erased.

Since enabling this step will permanently disable erase for UICR, the procedure require an implementation defined 32-bit word to be written into the `UICR->ERASEPROTECT` register.

In case of field return handling it is still possible to erase UICR even if `ERASEPROTECT` is set. If this functionality is desired, the secure boot code must implement a secure communication channel over the

CTRL-AP mailbox interface. Upon successful authentication of the external party, the secure boot code can temporarily re-enable the CTRL-AP ERASEALL functionality.

6.8.4 Usage

This section describe specific KMU and UICR behavior in more detail, in order to help the reader to get a better overview of its features and intended usage.

6.8.4.1 OTP

The OTP region of UICR contains user-defined static configuration of the device. The KMU emulates the OTP functionality by placing a write-once per halfword limitation of registers defined in this region, i.e. only halfwords containing all '1' can be written.

An OTP write transaction must consist of a full 32-bit word. Both halfwords can either be written simultaneously or individually. The KMU FSM will block any write to a halfword in the OTP region if the initial value of this half-word is not `0xFFFF`. When writing halfwords individually, the non-active halfword must be masked as `0xFFFF` else the request will be blocked. I.e. writing `0x1234XXXX` to an OTP destination address which already contain the value `0xFFFFAABB` must be configured as `0x1234FFFF`. The OTP destination address will contain the value `0x1234AABB` after both write transactions have been processed.

The KMU will also only allow AHB write transactions into the OTP region of UICR if the transaction is secure. Any AHB write transaction to this region that does not satisfy the above requirements will be ignored, and the `STATUS.BLOCKED` register will be set to '1'.

6.8.4.2 Key storage

The key storage region of UICR can contain multiple keys of different type, including symmetrical keys, hashes, public/private key pairs and other device secrets. One of the key features of the KMU, is that these device secrets can be installed and made available for use in cryptographic operations without revealing the actual secret values.

Keys in this region will typically have a certain life-span, and is not designed to be used for per-session derived keys where the non-secure side (i.e. application) is participating in the key exchange.

All key storage is done through the concept of multiple key slots, where one key slot instance consists of one key header and an associated key value. Each key header supports configuration of usage permissions and an optional secure destination address.

The key header secure destination address option enables the KMU to push the associated key value over a dedicated secure APB to a pre-configured secure location within the memory map. Such locations typically include write-only key register of a HW cryptograhic accelerator, allowing the KMU to distribute keys within the system without compromising the key values.

One key slot instance can store a key value of maximum 128 bits. If a key size exceeds this limit, the key value itself must be split over multiple key slot instances.

The following usage and read permissions scheme is applicable for each key slot:

State	Push	Read	Write	Description
Active (1)	Enabled (1)	Enabled (1)	Enabled (1)	Default flash erase value. Key slot cannot be pushed, write is enabled.
Active (1)	Enabled (1)	Enabled (1)	Disabled (0)	Key slot is active, push is enabled. Key slot VALUE registers can be read, but write is disabled.
Active (1)	Enabled (1)	Disabled (0)	Disabled (0)	Key slot is active, push is enabled. Read and write to key slot VALUE registers is disabled.
Active (1)	Disabled (0)	Enabled (1)	Disabled (0)	Key slot is active, push is disabled. Key slot VALUE registers can be read, but write is disabled.
Revoked (0)	-	-	-	Key slot is revoked. Cannot be read or pushed over Secure APB regardless of permission settings.

Table 48: Valid key slot permission schemes

6.8.4.2.1 Selecting a key slot

The KMU FSM is designed to process only one key slot at a time, effectively operating as a memory protection unit for the key storage region. Whenever a key slot is selected, the KMU will allow access to writing, reading, and/or pushing the associated key value according to the selected slot configuration.

A key slot **must** be selected prior to use by writing the key slot ID into the `KMU->SELECTKEYSLOT` register. Because the reset value of this register is `0x00000000`, there is no key slot associated with ID=0 and no slot is selected by default. All key slots are addressed using IDs from 1 to 128.

SELECTED status is set, when a key slot is selected and a read or write acccess to that keyslot occurs.

BLOCKED status is set, when any illegal access to key slot registers is detected.

When the use of the particular key slot is stopped, the key slot selection in `KMU->SELECTKEYSLOT` must be set back to '0'.

By default all KMU key slots will consist of a 128 bit key value of '1', where the key headers have no secure destination address or any usage and read restrictions.

6.8.4.2.2 Writing to a key slot

Writing a key slot into UICR is a five-step process.

1. Select which key slot the KMU shall operate on by writing the desired key slot `ID` into `KMU->SELECTKEYSLOT`. The selected key slot must be empty in order to add a new entry to UICR.
2. If the key value shall be pushable over secure APB, the destination address of the recipient must be configured in register `KEYSLOT.CONFIG[ID-1].DEST`.
3. Write the 128-bit key value into `KEYSLOT.KEY[ID-1].VALUE[0-3]`.
4. Write the desired key slot permissions into `KEYSLOT.CONFIG[ID-1].PERM`, including any applicable usage restrictions.
5. Select key slot `0`.

In case the total key size is greater than 128 bits, the key value itself must be split into 128-bit segments and written to multiple key slot instances. Steps 1 through 5 above must be repeated for the entire key size.

Note: If a key slot is configured as readable, and `KEYSLOT.CONFIG[ID-1].DEST` is not to be used, it is recommended to disable the push bit in `KEYSLOT.CONFIG[ID-1].PERM` when configuring key slot permissions.

Note: A key value distributed over multiple key slots should use the same key slot configuration in its key headers, but the secure destination address for each key slot instance must be incremented by 4 words (128 bits) for each key slot instance spanned.

Note: Write to flash must be enabled in `NVMC->CONFIG` prior to writing keys to flash, and subsequently disabled once writing is complete.

Steps 1 through 5 above will be blocked if any of the following violations are detected:

- No key slot selected
- Non-empty key slot selected
- NVM destination address not empty
- AHB write to `KEYSLOT.KEY[ID-1].VALUE[0-3]` registers not belonging to selected key slot

6.8.4.2.3 Reading a key value

Key slots that are configured as readable can have their key value read directly from the UICR memory map by the CPU.

Readable keys are typically used during the secure boot sequence, where the CPU is involved in falsifying or verifying the integrity of the system. Since the CPU is involved in this decision process, it makes little sense not to trust the CPU having access to actual key value but ultimately trust the decision of the integrity check. Another use-case for readable keys is if the key type in question does not have a HW peripheral in the platform that is able to accept such keys over secure APB.

Reading a key value from UICR is a three-step process:

1. Select the key slot which the KMU shall operate on by writing the desired key slot `ID` into `KMU->SELECTKEYSLOT`.
2. If STATE and READ permission requirements are fulfilled as defined in `KEYSLOT.CONFIG[ID-1].PERM`, the key value can be read from region `KEYSLOT.KEY[ID-1].VALUE[0-3]` for selected key slot.
3. Select key slot `0`.

Step 2 will be blocked and word `0xDEADDEAD` will be returned on AHB if any of the following violations are detected:

- No key slot selected
- Key slot not configured as readable
- Key slot is revoked
- AHB read to `KEYSLOT.KEY[ID-1].VALUE[0-3]` registers not belonging to selected key slot

6.8.4.2.4 Push over secure APB

Key slots that are configured as non-readable cannot be read by the CPU regardless of mode the system is in, and must be pushed over secure APB in order to use the key value for cryptographic operations.

The secure APB destination address is set in the key slot configuration DEST register. Such destination addresses are typically write-only key registers in a hardware cryptographic accelerators memory map. The secure APB allows key slots to be utilized by the software side, without exposing the key value itself.

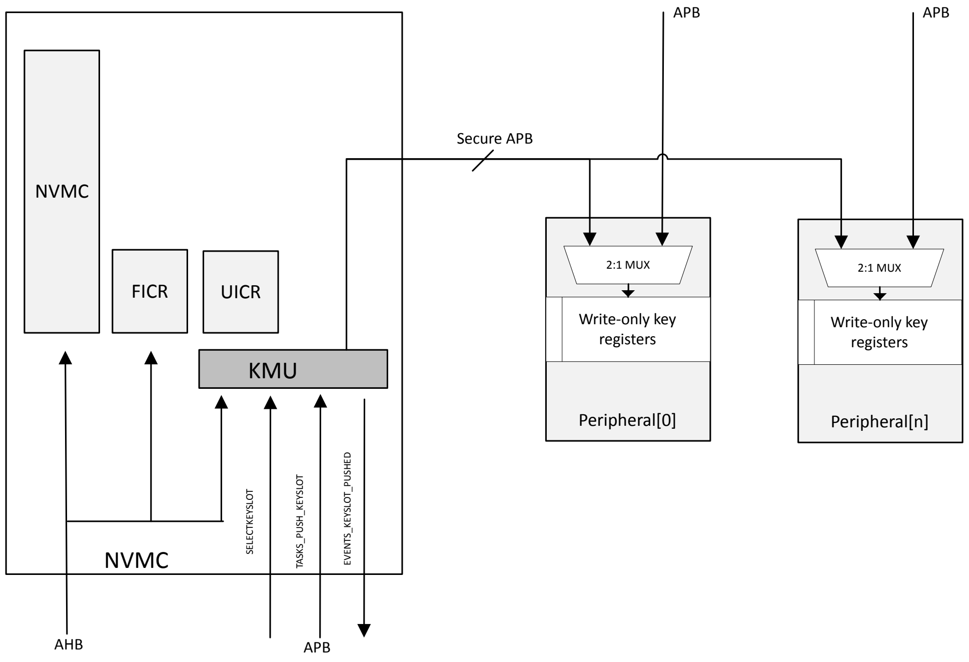


Figure 33: Tasks and events pattern for key slots

Pushing a key slot over secure APB is a four-step process:

1. Select the key slot on which the KMU shall operate by writing the desired key slot `ID` into `KMU->SELECTKEYSLOT`
2. Start `TASKS_PUSH_KEYSLOT` to initiate a secure APB transaction writing the 128-bit key value associated with the selected key slot into address defined in `KEYSLOT.CONFIG[ID-1].DEST`
3. After completing the secure APB transaction, the 128-bit key value is ready for use by the peripheral and `EVENTS_KEYSLOT_PUSHED` is triggered
4. Select key slot `0`

Note: If a key value is distributed over multiple key slots due to its key size, exceeding the maximum 128-bit key value limitation, then each distributed key slot must be pushed individually in order to transfer the entire key value over secure APB.

Step 3 will trigger other events than `EVENTS_KEYSLOT_PUSHED` if the following violations are detected:

- `EVENTS_KEYSLOT_ERROR`:
  - If no key slot is selected
  - If a key slot has no destination address configured
  - If when pushing a key slot, flash or peripheral returns an error
  - If pushing a key slot when push permissions are disabled
  - If attempting to push a key slot with default permissions
- `EVENTS_KEYSLOT_REVOKED` if a key slot is marked as revoked in its key header configuration

6.8.4.2.5 Revoking key slots

All key slots within the key storage area can be marked as revoked by writing to the `STATE` field in the `KEYSLOT.CONFIG[ID-1].PERM` register. The following rules apply to keys that have been revoked:

1. Key values that are not readable by the CPU, and thus depend on the tasks/events pattern to be used by a peripheral, can no longer be pushed. If a revoked key slot is selected and task `TASKS_PUSH_KEYSLOT` is started, the event `EVENTS_KEYSLOT_REVOKED` will be triggered.
2. Key values that are readable by the CPU can have their revoke bit set in order to instruct the KMU to block future read requests for this key value. Any subsequent read operation to a revoked key value will return word `0xDEADDEAD`.
3. Published keys stored in a peripheral write-only key register are not affected by key revocation. If secure code wants to enforce that a revoked key is no longer used by a peripheral for cryptographic operations, the secure code need to reset the device and thus prevent the revoked key slot from being published again.

6.8.4.3 STATUS register

The KMU uses a `KMU->STATUS` register to indicate its status of operation. The `SELECTED` bit will be asserted whenever the currently selected key slot is successfully read from or written to.

All read or write operations to other key slots than what is currently selected in `KMU->SELECTKEYSLOT` will assert the `BLOCKED` bit. The `BLOCKED` bit will also be asserted if the KMU fails to select a key slot, or if a request has been blocked due to an access violation. Normal operation using the KMU should never trigger the `BLOCKED` bit. If this bit is triggered during the development phase, this indicate that code is using the KMU incorrectly.

The `KMU->STATUS` register is reset every time register `KMU->SELECTKEYSLOT` is written.

6.8.5 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50039000 0x40039000	KMU	KMU : S KMU : NS	SPLIT	NA	Key management unit	

Table 49: Instances

Register	Offset	Security	Description
TASKS_PUSH_KEYSLOT	0x0000		Push a key slot over secure APB
EVENTS_KEYSLOT_PUSHED	0x100		Key successfully pushed over secure APB
EVENTS_KEYSLOT_REVOKED	0x104		Key has been revoked and cannot be tasked for selection
EVENTS_KEYSLOT_ERROR	0x108		No key slot selected, no destination address defined, or error during push operation
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
INTPEND	0x30C		Pending interrupts
STATUS	0x40C		Status bits for KMU operation
SELECTKEYSLOT	0x500		Select key slot ID to be read over AHB or pushed over secure APB when TASKS_PUSH_KEYSLOT is started

Table 50: Register overview

6.8.5.1 TASKS_PUSH_KEYSLOT

Address offset: 0x0000

Push a key slot over secure APB

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_PUSH_KEYSLOT			Push a key slot over secure APB
			Trigger	1	Trigger task

6.8.5.2 EVENTS_KEYSLOT_PUSHED

Address offset: 0x100

Key successfully pushed over secure APB

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_KEYSLOT_PUSHED			Key successfully pushed over secure APB
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.8.5.3 EVENTS_KEYSLOT_REVOKED

Address offset: 0x104

Key has been revoked and cannot be tasked for selection

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_KEYSLOT_REVOKED			Key has been revoked and cannot be tasked for selection
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.8.5.4 EVENTS_KEYSLOT_ERROR

Address offset: 0x108

No key slot selected, no destination address defined, or error during push operation

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_KEYSLOT_ERROR			No key slot selected, no destination address defined, or error during push operation
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.8.5.5 INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	KEYSLOT_PUSHED			Enable or disable interrupt for event KEYSLOT_PUSHED
			Disabled	0	Disable
			Enabled	1	Enable
B	RW	KEYSLOT_REVOKED			Enable or disable interrupt for event KEYSLOT_REVOKED
			Disabled	0	Disable
			Enabled	1	Enable
C	RW	KEYSLOT_ERROR			Enable or disable interrupt for event KEYSLOT_ERROR
			Disabled	0	Disable
			Enabled	1	Enable

6.8.5.6 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	KEYSLOT_PUSHED			Write '1' to enable interrupt for event KEYSLOT_PUSHED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	KEYSLOT_REVOKED			Write '1' to enable interrupt for event KEYSLOT_REVOKED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	KEYSLOT_ERROR			Write '1' to enable interrupt for event KEYSLOT_ERROR
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.8.5.7 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	KEYSLOT_PUSHED			Write '1' to disable interrupt for event KEYSLOT_PUSHED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	KEYSLOT_REVOKED			Write '1' to disable interrupt for event KEYSLOT_REVOKED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
C	RW	KEYSLOT_ERROR			Write '1' to disable interrupt for event KEYSLOT_ERROR
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.8.5.8 INTPEND

Address offset: 0x30C

Pending interrupts

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	KEYSLOT_PUSHED			Read pending status of interrupt for event KEYSLOT_PUSHED
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
B	R	KEYSLOT_REVOKED			Read pending status of interrupt for event KEYSLOT_REVOKED
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending
C	R	KEYSLOT_ERROR			Read pending status of interrupt for event KEYSLOT_ERROR
			NotPending	0	Read: Not pending
			Pending	1	Read: Pending

6.8.5.9 STATUS

Address offset: 0x40C

Status bits for KMU operation

This register is reset and re-written by the KMU whenever SELECTKEYSLOT is written

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	SELECTED			Key slot ID successfully selected by the KMU
			Disabled	0	No key slot ID selected by KMU
			Enabled	1	Key slot ID successfully selected by KMU
B	R	BLOCKED			Violation status
			Disabled	0	No access violation detected
			Enabled	1	Access violation detected and blocked

6.8.5.10 SELECTKEYSLOT

Address offset: 0x500

Select key slot ID to be read over AHB or pushed over secure APB when TASKS_PUSH_KEYSLOT is started

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	ID			Select key slot ID to be read over AHB, or pushed over secure APB, when TASKS_PUSH_KEYSLOT is started NOTE: ID=0 is not a valid key ID. The 0 ID should be used when the KMU is idle or not in use NOTE: Note that index N in UICR->KEYSLOT.KEY[N] and UICR->KEYSLOT.CONFIG[N] corresponds to KMU keyslot ID=N+1

6.9 PCGCMASTER — Power and clock master backdoors

The PCGCMASTER provides backdoor registers for a PCGC Master.

TODO.

6.10 PCGCSLAVE — Power and clock slave backdoors

The PCGCSLAVE provides backdoor registers for the PCGC control of peripherals and modules.

TODO.

6.11 PDM — Pulse density modulation interface

The pulse density modulation (PDM) module enables input of pulse density modulated signals from external audio frontends, for example, digital microphones. The PDM module generates the PDM clock and supports single-channel or dual-channel (Left and Right) data input. Data is transferred directly to RAM buffers using EasyDMA.

Listed here are the main features for PDM:

- Up to two PDM microphones configured as a Left/Right pair using the same data input
- 16 kHz output sample rate, 16-bit samples
- EasyDMA support for sample buffering
- HW decimation filters
- Selectable ratio of 64 or 80 between PDM_CLK and output sample rate

The PDM module illustrated in PDM module on page 146 is interfacing up to two digital microphones with the PDM interface. It implements EasyDMA, which relieves real-time requirements associated with controlling the PDM slave from a low priority CPU execution context. It also includes all the necessary digital filter elements to produce PCM samples. The PDM module allows continuous audio streaming.

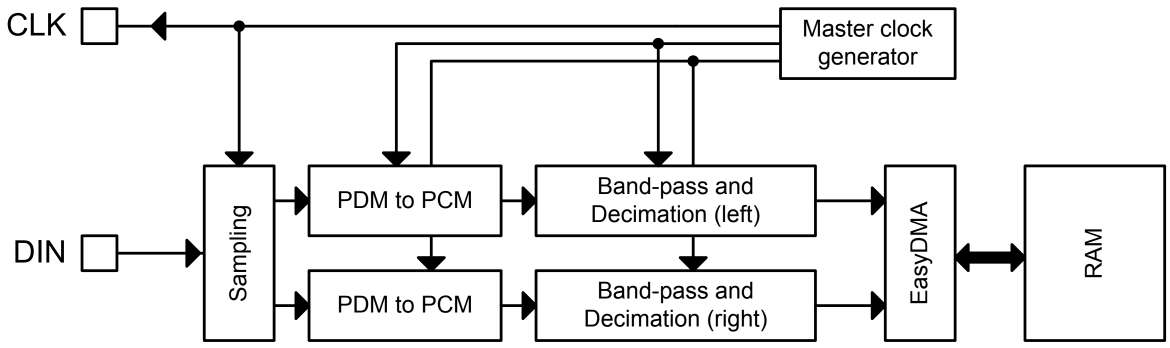


Figure 34: PDM module

6.11.1 Master clock generator

The FREQ field in the master clock's PDMCLKCTRL register allows adjusting the PDM clock's frequency.

The master clock generator does not add any jitter to the HFCLK source chosen. It is recommended (but not mandatory) to use the Xtal as HFCLK source.

6.11.2 Module operation

By default, bits from the left PDM microphone are sampled on PDM_CLK falling edge, bits for the right are sampled on the rising edge of PDM_CLK, resulting in two bitstreams. Each bitstream is fed into a digital filter which converts the PDM stream into 16-bit PCM samples, and filters and down-samples them to reach the appropriate sample rate.

The EDGE field in the MODE register allows swapping Left and Right, so that Left will be sampled on rising edge, and Right on falling.

The PDM module uses EasyDMA to store the samples coming out from the filters into one buffer in RAM.

Depending on the mode chosen in the OPERATION field in the MODE register, memory either contains alternating left and right 16-bit samples (Stereo), or only left 16-bit samples (Mono).

To ensure continuous PDM sampling, it is up to the application to update the EasyDMA destination address pointer as the previous buffer is filled.

The continuous transfer can be started or stopped by sending the START and STOP tasks. STOP becomes effective after the current frame has finished transferring, which will generate the STOPPED event. The STOPPED event indicates that all activity in the module are finished, and that the data is available in RAM (EasyDMA has finished transferring as well). Attempting to restart before receiving the STOPPED event may result in unpredictable behaviour.

6.11.3 Decimation filter

In order to convert the incoming data stream into PCM audio samples, a decimation filter is included in the PDM interface module.

The input of the filter is the two-channel PDM serial stream (with left channel on clock high, right channel on clock low). Depending on the RATIO selected, its output is 2 × 16-bit PCM samples at a sample rate either 64 times or 80 times (depending on the RATIO register) lower than the PDM clock rate.

The filter stage of each channel is followed by a digital volume control, to attenuate or amplify the output samples in a range of -20 dB to +20 dB around the default (reset) setting, defined by $G_{PDM,default}$. The gain is controlled by the GAINL and GAINR registers.

As an example, if the goal is to achieve 2500 RMS output samples (16 bit) with a 1 kHz 90 dBA signal into a -26 dBFS sensitivity PDM microphone, the user will have to sum the PDM module's default gain ($G_{PDM,default}$) and the gain introduced by the microphone and acoustic path of his implementation (an attenuation would translate into a negative gain), and adjust GAINL and GAINR by this amount. Assuming that only the PDM module influences the gain, GAINL and GAINR must be set to -$G_{PDM,default}$ dB to achieve the requirement.

With $G_{PDM,default}$=3.2 dB, and as GAINL and GAINR are expressed in 0.5 dB steps, the closest value to program would be 3.0 dB, which can be calculated as:

```
GAINL = GAINR = (DefaultGain - (2 * 3))
```

Remember to check that the resulting values programmed into GAINL and GAINR fall within MinGain and MaxGain.

6.11.4 EasyDMA

Samples will be written directly to RAM, and EasyDMA must be configured accordingly.

The address pointer for the EasyDMA channel is set in SAMPLE.PTR register. If the destination address set in SAMPLE.PTR is not pointing to the Data RAM region, an EasyDMA transfer may result in a HardFault or RAM corruption. See Memory on page 20 for more information about the different memory regions.

DMA supports Stereo (Left+Right 16-bit samples) and Mono (Left only) data transfer, depending on setting in the OPERATION field in the MODE register. The samples are stored little endian.

MODE.OPERATION	Bits per sample	Result stored per RAM word	Physical RAM allocated (32 bit words)	Result boundary indexes in RAM	Note
Stereo	32 (2x16)	L+R	ceil(SAMPLE.MAXCNT/2)	R0=[31:16]; L0=[15:0]	Default
Mono	16	2xL	ceil(SAMPLE.MAXCNT/2)	L1=[31:16]; L0=[15:0]	

Table 51: DMA sample storage

The destination buffer in RAM consists of one block, the size of which is set in SAMPLE.MAXCNT register. Format is number of 16-bit samples. The physical RAM allocated is always:

```
      (RAM allocation, in bytes) = SAMPLE.MAXCNT * 2;
```

(but the mapping of the samples depends on MODE.OPERATION.

If OPERATION=Stereo, RAM will contain a succession of Left and Right samples.

If OPERATION=Mono, RAM will contain a succession of mono samples.

For a given value of SAMPLE.MAXCNT, the buffer in RAM can contain half the stereo sampling time as compared to the mono sampling time.

The PDM acquisition can be started by the START task, after the SAMPLE.PTR and SAMPLE.MAXCNT registers have been written. When starting the module, it will take some time for the filters to start outputting valid data. Transients from the PDM microphone itself may also occur. The first few samples (typically around 50) might hence contain invalid values or transients. It is therefore advised to discard the first few samples after a PDM start.

As soon as the STARTED event is received, the firmware can write the next SAMPLE.PTR value (this register is double-buffered), to ensure continuous operation.

When the buffer in RAM is filled with samples, an END event is triggered. The firmware can start processing the data in the buffer. Meanwhile, the PDM module starts acquiring data into the new buffer pointed to by SAMPLE.PTR, and sends a new STARTED event, so that the firmware can update SAMPLE.PTR to the next buffer address.

6.11.5 Hardware example

Connect the microphone clock to CLK, and data to DIN.

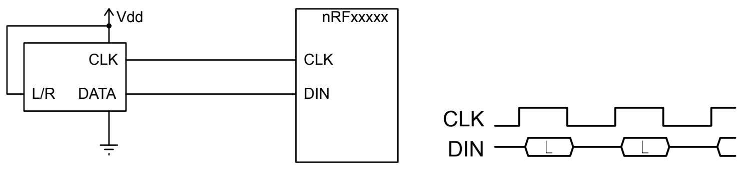


Figure 35: Example of a single PDM microphone, wired as left

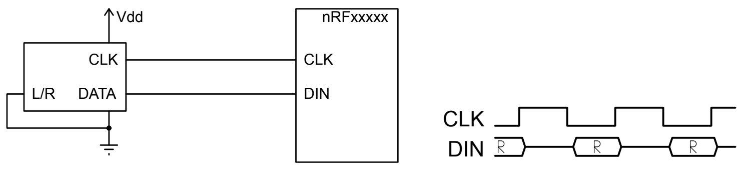


Figure 36: Example of a single PDM microphone, wired as right

Note that in a single-microphone (mono) configuration, depending on the microphone's implementation, either the left or the right channel (sampled at falling or rising CLK edge respectively) will contain reliable data. If two microphones are used, one of them has to be set as left, the other as right (L/R pin tied high or to GND on the respective microphone). It is strongly recommended to use two microphones of exactly the same brand and type so that their timings in left and right operation match.

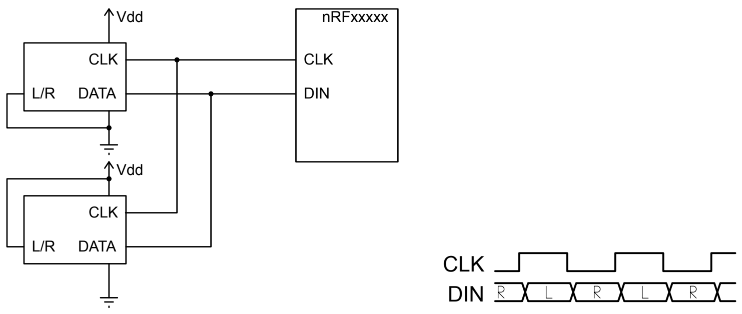


Figure 37: Example of two PDM microphones

6.11.6 Pin configuration

The CLK and DIN signals associated to the PDM module are mapped to physical pins according to the configuration specified in the PSEL.CLK and PSEL.DIN registers respectively. If the CONNECT field in any PSEL register is set to Disconnected, the associated PDM module signal will not be connected to the required physical pins, and will not operate properly.

The PSEL.CLK and PSEL.DIN registers and their configurations are only used as long as the PDM module is enabled, and retained only as long as the device is in System ON mode. See POWER — Power control on page 58 for more information about power modes. When the peripheral is disabled, the pins will behave as regular GPIOs, and use the configuration in their respective OUT bit field and PIN_CNF[n] register.

To ensure correct behaviour in the PDM module, the pins used by the PDM module must be configured in the GPIO peripheral as described in GPIO configuration before enabling peripheral on page 149 before enabling the PDM module. This is to ensure that the pins used by the PDM module are driven correctly if the PDM module itself is temporarily disabled or the device temporarily enters System OFF. This configuration must be retained in the GPIO for the selected I/Os as long as the PDM module is supposed to be connected to an external PDM circuit.

Only one peripheral can be assigned to drive a particular GPIO pin at a time. Failing to do so may result in unpredictable behaviour.

PDM signal	PDM pin	Direction	Output value	Comment
CLK	As specified in PSEL.CLK	Output	0	
DIN	As specified in PSEL.DIN	Input	Not applicable	

Table 52: GPIO configuration before enabling peripheral

6.11.7 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50026000 0x40026000	PDM	PDM : S PDM : NS	US	SA	Pulse density modulation (digital microphone) interface	

Table 53: Instances

Register	Offset	Security	Description
TASKS_START	0x000		Starts continuous PDM transfer
TASKS_STOP	0x004		Stops PDM transfer
SUBSCRIBE_START	0x080		Subscribe configuration for task START
SUBSCRIBE_STOP	0x084		Subscribe configuration for task STOP
EVENTS_STARTED	0x100		PDM transfer has started
EVENTS_STOPPED	0x104		PDM transfer has finished
EVENTS_END	0x108		The PDM has written the last sample specified by SAMPLE.MAXCNT (or the last sample after a STOP task has been received) to Data RAM
PUBLISH_STARTED	0x180		Publish configuration for event STARTED
PUBLISH_STOPPED	0x184		Publish configuration for event STOPPED
PUBLISH_END	0x188		Publish configuration for event END
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
ENABLE	0x500		PDM module enable register
PDMCLKCTRL	0x504		PDM clock generator control
MODE	0x508		Defines the routing of the connected PDM microphones' signals
GAINL	0x518		Left output gain adjustment
GAINR	0x51C		Right output gain adjustment
RATIO	0x520		Selects the ratio between PDM_CLK and output sample rate. Change PDMCLKCTRL accordingly.
PSEL.CLK	0x540		Pin number configuration for PDM CLK signal
PSEL.DIN	0x544		Pin number configuration for PDM DIN signal
SAMPLE.PTR	0x560		RAM address pointer to write samples to with EasyDMA
SAMPLE.MAXCNT	0x564		Number of samples to allocate memory for in EasyDMA mode

Table 54: Register overview

6.11.7.1 TASKS_START

Address offset: 0x000

Starts continuous PDM transfer

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_START			Starts continuous PDM transfer
			Trigger	1	Trigger task

6.11.7.2 TASKS_STOP

Address offset: 0x004

Stops PDM transfer

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_STOP			Stops PDM transfer
			Trigger	1	Trigger task

6.11.7.3 SUBSCRIBE_START

Address offset: 0x080

Subscribe configuration for task START

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task START will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.11.7.4 SUBSCRIBE_STOP

Address offset: 0x084

Subscribe configuration for task STOP

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task STOP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.11.7.5 EVENTS_STARTED

Address offset: 0x100

PDM transfer has started

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_STARTED			PDM transfer has started
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.11.7.6 EVENTS_STOPPED

Address offset: 0x104

PDM transfer has finished

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_STOPPED			PDM transfer has finished
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.11.7.7 EVENTS_END

Address offset: 0x108

The PDM has written the last sample specified by SAMPLE.MAXCNT (or the last sample after a STOP task has been received) to Data RAM

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_END			The PDM has written the last sample specified by SAMPLE.MAXCNT (or the last sample after a STOP task has been received) to Data RAM
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.11.7.8 PUBLISH_STARTED

Address offset: 0x180

Publish configuration for event STARTED

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event STARTED will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.11.7.9 PUBLISH_STOPPED

Address offset: 0x184

Publish configuration for event STOPPED

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event STOPPED will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.11.7.10 PUBLISH_END

Address offset: 0x188

Publish configuration for event END

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event END will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.11.7.11 INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	STARTED			Enable or disable interrupt for event STARTED
			Disabled	0	Disable
			Enabled	1	Enable
B	RW	STOPPED			Enable or disable interrupt for event STOPPED
			Disabled	0	Disable
			Enabled	1	Enable
C	RW	END			Enable or disable interrupt for event END
			Disabled	0	Disable
			Enabled	1	Enable

6.11.7.12 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	STARTED			Write '1' to enable interrupt for event STARTED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	STOPPED			Write '1' to enable interrupt for event STOPPED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	END			Write '1' to enable interrupt for event END
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.11.7.13 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	STARTED			Write '1' to disable interrupt for event STARTED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	STOPPED			Write '1' to disable interrupt for event STOPPED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C	RW	END			Write '1' to disable interrupt for event END
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.11.7.14 ENABLE

Address offset: 0x500

PDM module enable register

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	ENABLE			Enable or disable PDM module
			Disabled	0	Disable
			Enabled	1	Enable

6.11.7.15 PDMCLKCTRL

Address offset: 0x504

PDM clock generator control

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x08400000	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	FREQ			PDM_CLK frequency
			1000K	0x08000000	PDM_CLK = 32 MHz / 32 = 1.000 MHz
			Default	0x08400000	PDM_CLK = 32 MHz / 31 = 1.032 MHz. Nominal clock for RATIO=Ratio64.
			1067K	0x08800000	PDM_CLK = 32 MHz / 30 = 1.067 MHz
			1231K	0x09800000	PDM_CLK = 32 MHz / 26 = 1.231 MHz
			1280K	0x0A000000	PDM_CLK = 32 MHz / 25 = 1.280 MHz. Nominal clock for RATIO=Ratio80.
			1333K	0x0A800000	PDM_CLK = 32 MHz / 24 = 1.333 MHz

6.11.7.16 MODE

Address offset: 0x508

Defines the routing of the connected PDM microphones' signals

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	OPERATION			Mono or stereo operation
			Stereo	0	Sample and store one pair (Left + Right) of 16bit samples per RAM word R=[31:16]; L=[15:0]
			Mono	1	Sample and store two successive Left samples (16 bit each) per RAM word L1=[31:16]; L0=[15:0]
B	RW	EDGE			Defines on which PDM_CLK edge Left (or mono) is sampled
			LeftFalling	0	Left (or mono) is sampled on falling edge of PDM_CLK
			LeftRising	1	Left (or mono) is sampled on rising edge of PDM_CLK

6.11.7.17 GAINL

Address offset: 0x518

Left output gain adjustment

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										A	A	A	A	A	A	A
Reset 0x00000028	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	GAINL			Left output gain adjustment, in 0.5 dB steps, around the default module gain (see electrical parameters) 0x00 -20 dB gain adjust 0x01 -19.5 dB gain adjust (...) 0x27 -0.5 dB gain adjust 0x28 0 dB gain adjust 0x29 +0.5 dB gain adjust (...) 0x4F +19.5 dB gain adjust 0x50 +20 dB gain adjust
			MinGain	0x00	-20dB gain adjustment (minimum)
			DefaultGain	0x28	0dB gain adjustment
			MaxGain	0x50	+20dB gain adjustment (maximum)

6.11.7.18 GAINR

Address offset: 0x51C

Right output gain adjustment

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																										A	A	A	A	A	A	A
Reset 0x00000028	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	GAINR			Right output gain adjustment, in 0.5 dB steps, around the default module gain (see electrical parameters)
			MinGain	0x00	-20dB gain adjustment (minimum)
			DefaultGain	0x28	0dB gain adjustment
			MaxGain	0x50	+20dB gain adjustment (maximum)

6.11.7.19 RATIO

Address offset: 0x520

Selects the ratio between PDM_CLK and output sample rate. Change PDMCLKCTRL accordingly.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	RATIO			Selects the ratio between PDM_CLK and output sample rate
			Ratio64	0	Ratio of 64
			Ratio80	1	Ratio of 80

6.11.7.20 PSEL.CLK

Address offset: 0x540

Pin number configuration for PDM CLK signal

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C																											A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

6.11.7.21 PSEL.DIN

Address offset: 0x544

Pin number configuration for PDM DIN signal

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C																											A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

6.11.7.22 SAMPLE.PTR

Address offset: 0x560

RAM address pointer to write samples to with EasyDMA

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	SAMPLEPTR			Address to write PDM samples to over DMA **Note:** See the memory chapter for details about which memories are available for EasyDMA.

6.11.7.23 SAMPLE.MAXCNT

Address offset: 0x564

Number of samples to allocate memory for in EasyDMA mode

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																		A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	BUFFSIZE		[0..32767]	Length of DMA RAM allocation in number of samples

6.11.8 Electrical specification

6.11.8.1 PDM Electrical Specification

Symbol	Description	Min.	Typ.	Max.	Units
$f_{PDM,CLK,64}$	PDM clock speed. PDMCLKCTRL = Default (Setting needed for 16MHz sample frequency @ RATIO = Ratio64)		1.032		MHz
$f_{PDM,CLK,80}$	PDM clock speed. PDMCLKCTRL = 1280K (Setting needed for 16MHz sample frequency @ RATIO = Ratio80)	..	..	..	MHz
$t_{PDM,JITTER}$	Jitter in PDM clock output			20	ns
$T_{dPDM,CLK}$	PDM clock duty cycle	40	50	60	%
$t_{PDM,DATA}$	Decimation filter delay			5	ms
$t_{PDM,cv}$	Allowed clock edge to data valid			125	ns
$t_{PDM,ci}$	Allowed (other) clock edge to data invalid	0			ns
$t_{PDM,s}$	Data setup time at $f_{PDM,CLK}$=1.024 MHz or 1.280 MHz	65			ns
$t_{PDM,h}$	Data hold time at $f_{PDM,CLK}$=1.024 MHz or 1.280 MHz	0			ns
$G_{PDM,default}$	Default (reset) absolute gain of the PDM module		3.2		dB

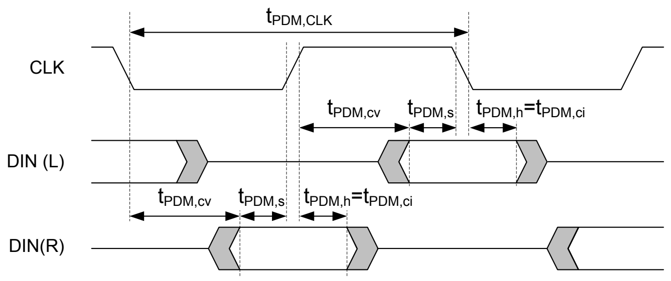


Figure 38: PDM timing diagram

6.12 PWM — Pulse width modulation

The pulse with modulation (PWM) module enables the generation of pulse width modulated signals on GPIO. The module implements an up or up-and-down counter with four PWM channels that drive assigned GPIOs.

The following are the main features of a PWM module:

- Programmable PWM frequency
- Up to four PWM channels with individual polarity and duty cycle values
- Edge or center-aligned pulses across PWM channels
- Multiple duty cycle arrays (sequences) defined in RAM
- Autonomous and glitch-free update of duty cycle values directly from memory through EasyDMA (no CPU involvement)
- Change of polarity, duty cycle, and base frequency possibly on every PWM period
- RAM sequences can be repeated or connected into loops

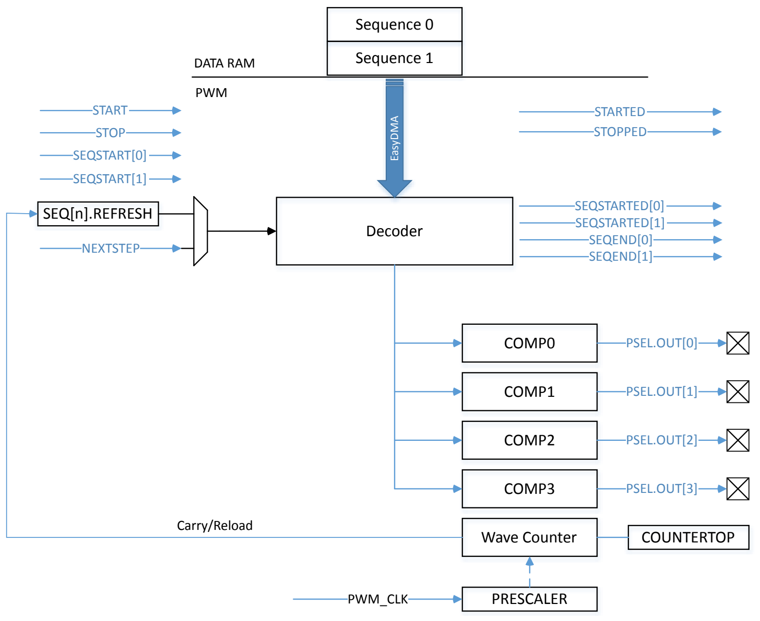


Figure 39: PWM module

6.12.1 Wave counter

The wave counter is responsible for generating the pulses at a duty cycle that depends on the compare values, and at a frequency that depends on COUNTERTOP.

There is one common 15-bit counter with four compare channels. Thus, all four channels will share the same period (PWM frequency), but can have individual duty cycle and polarity. The polarity is set by a value read from RAM (see figure Decoder memory access modes on page 161). Whether the counter counts up, or up and down, is controlled by the MODE register.

The timer top value is controlled by the COUNTERTOP register. This register value, in conjunction with the selected PRESCALER of the PWM_CLK, will result in a given PWM period. A COUNTERTOP value smaller than the compare setting will result in a state where no PWM edges are generated. OUT[n] is held high, given that the polarity is set to FallingEdge. All compare registers are internal and can only be configured through decoder presented later. COUNTERTOP can be safely written at any time.

Sampling follows the START task. If DECODER.LOAD=WaveForm, the register value is ignored and taken from RAM instead (see section Decoder with EasyDMA on page 161 for more details). If DECODER.LOAD is anything else than the WaveForm, it is sampled following a STARTSEQ[n] task and when loading a new value from RAM during a sequence playback.

The following figure shows the counter operating in up mode (MODE=PWM_MODE_Up), with three PWM channels with the same frequency but different duty cycle:

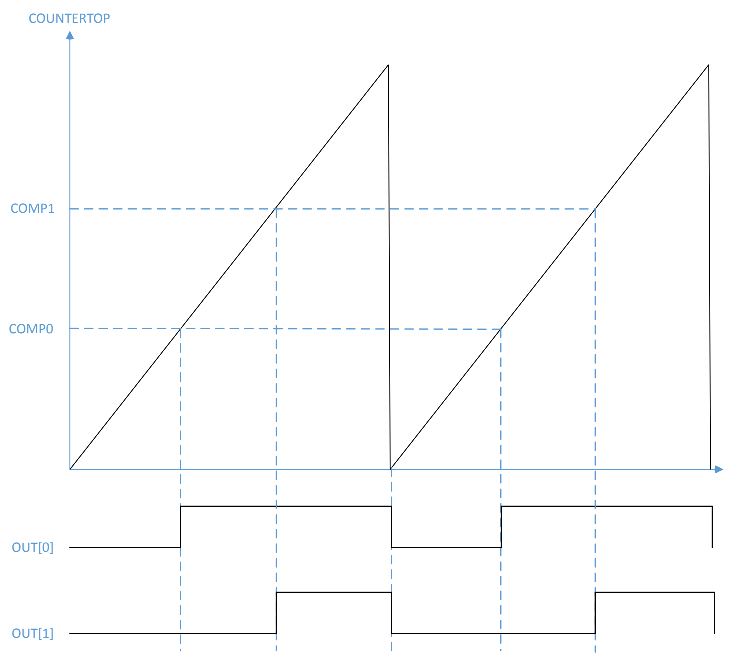


Figure 40: PWM counter in up mode example - FallingEdge polarity

The counter is automatically reset to zero when COUNTERTOP is reached and OUT[n] will invert. OUT[n] is held low if the compare value is 0 and held high if set to COUNTERTOP, given that the polarity is set to FallingEdge. Counter running in up mode results in pulse widths that are edge-aligned. The following is the code for the counter in up mode example:

```
uint16_t pwm_seq[4] = {PWM_CH0_DUTY, PWM_CH1_DUTY, PWM_CH2_DUTY, PWM_CH3_DUTY};
NRF_PWM0->PSEL.OUT[0] = (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                                 PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->PSEL.OUT[1] = (second_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                                 PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_Up << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                                 PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
NRF_PWM0->LOOP        = (PWM_LOOP_CNT_Disabled << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER   = (PWM_DECODER_LOAD_Individual << PWM_DECODER_LOAD_Pos) |
                      (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
NRF_PWM0->SEQ[0].PTR  = ((uint32_t)(pwm_seq) << PWM_SEQ_PTR_PTR_Pos);
NRF_PWM0->SEQ[0].CNT  = ((sizeof(pwm_seq) / sizeof(uint16_t)) <<
                                                 PWM_SEQ_CNT_CNT_Pos);
NRF_PWM0->SEQ[0].REFRESH  = 0;
NRF_PWM0->SEQ[0].ENDDELAY = 0;
NRF_PWM0->TASKS_SEQSTART[0] = 1;
```

When the counter is running in up mode, the following formula can be used to compute the PWM period and the step size:

PWM period: $T_{PWM(Up)} = T_{PWM\_CLK} * \mathrm{COUNTERTOP}$

Step width/Resolution: $T_{steps} = T_{PWM\_CLK}$

The following figure shows the counter operating in up-and-down mode (MODE=PWM_MODE_UpAndDown), with two PWM channels with the same frequency but different duty cycle and output polarity:

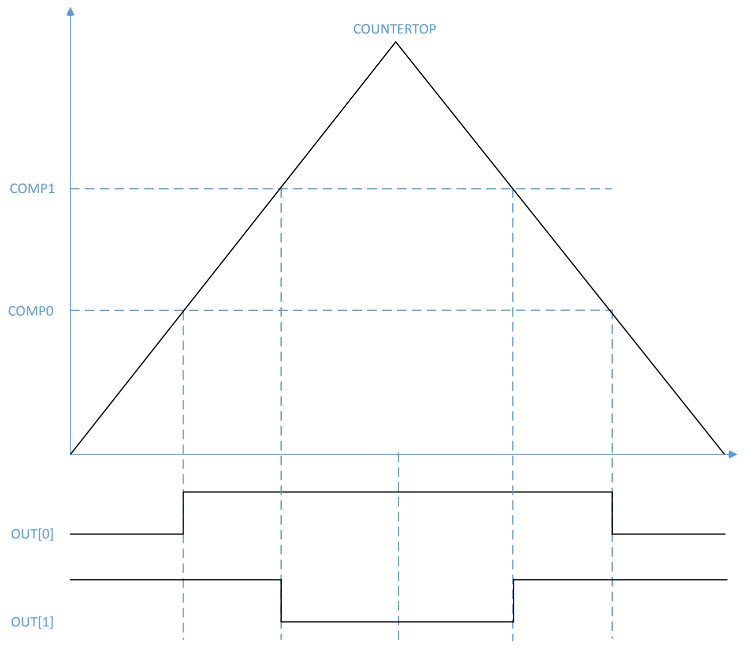


Figure 41: PWM counter in up-and-down mode example

The counter starts decrementing to zero when COUNTERTOP is reached and will invert the OUT[n] when compare value is hit for the second time. This results in a set of pulses that are center-aligned. The following is the code for the counter in up-and-down mode example:

```
uint16_t pwm_seq[4] = {PWM_CH0_DUTY, PWM_CH1_DUTY, PWM_CH2_DUTY, PWM_CH3_DUTY};
NRF_PWM0->PSEL.OUT[0] = (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                                 PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->PSEL.OUT[1] = (second_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                                 PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_UpAndDown << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                                 PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
NRF_PWM0->LOOP        = (PWM_LOOP_CNT_Disabled << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER    = (PWM_DECODER_LOAD_Individual << PWM_DECODER_LOAD_Pos) |
                       (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
NRF_PWM0->SEQ[0].PTR  = ((uint32_t)(pwm_seq) << PWM_SEQ_PTR_PTR_Pos);
NRF_PWM0->SEQ[0].CNT  = ((sizeof(pwm_seq) / sizeof(uint16_t)) <<
                                                 PWM_SEQ_CNT_CNT_Pos);
NRF_PWM0->SEQ[0].REFRESH  = 0;
NRF_PWM0->SEQ[0].ENDDELAY = 0;
NRF_PWM0->TASKS_SEQSTART[0] = 1;
```

When the counter is running in up-and-down mode, the following formula can be used to compute the PWM period and the step size:

$$T_{PWM(Up\ And\ Down)} = T_{PWM\_CLK} * 2 * COUNTERTOP$$

Step width/Resolution: $T_{steps} = T_{PWM\_CLK} * 2$

6.12.2 Decoder with EasyDMA

The decoder uses EasyDMA to take PWM parameters stored in RAM and update the internal compare registers of the wave counter, based on the mode of operation.

PWM parameters are organized into a sequence containing at least one half word (16 bit). Its most significant bit[15] denotes the polarity of the OUT[n] while bit[14:0] is the 15-bit compare value.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Id																	B	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Id	RW	Field	Value Id	Value	Description
A	RW	COMPARE			Duty cycle setting - value loaded to internal compare register
B	RW	POLARITY			Edge polarity of GPIO.
			RisingEdge	0	First edge within the PWM period is rising
			FallingEdge	1	First edge within the PWM period is falling

The DECODER register controls how the RAM content is interpreted and loaded into the internal compare registers. The LOAD field controls if the RAM values are loaded to all compare channels, or to update a group or all channels with individual values. The following figure illustrates how parameters stored in RAM are organized and routed to various compare channels in different modes:

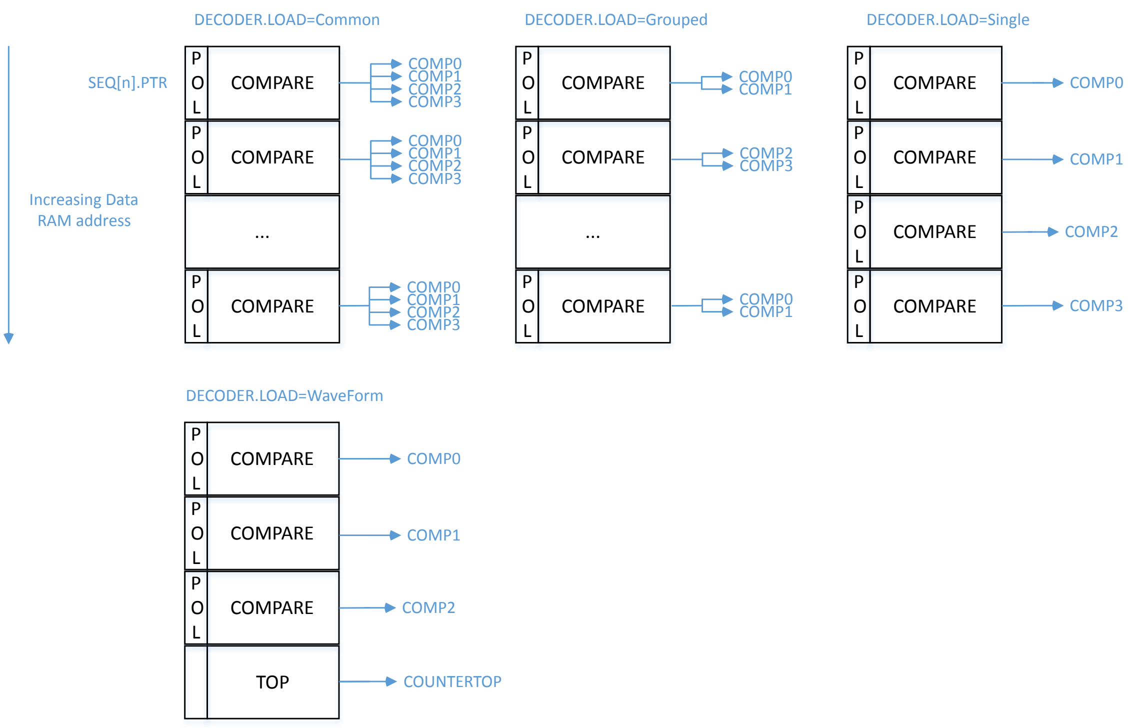


Figure 42: Decoder memory access modes

A special mode of operation is available when DECODER.LOAD is set to WaveForm. In this mode, up to three PWM channels can be enabled - OUT[0] to OUT[2]. In RAM, four values are loaded at a time: the first, second and third location are used to load the values, and the fourth RAM location is used to load the COUNTERTOP register. This way one can have up to three PWM channels with a frequency base that changes on a per PWM period basis. This mode of operation is useful for arbitrary wave form generation in applications, such as LED lighting.

The register SEQ[n].REFRESH=N (one per sequence n=0 or 1) will instruct a new RAM stored pulse width value on every (N+1)th PWM period. Setting the register to zero will result in a new duty cycle update every PWM period, as long as the minimum PWM period is observed.

Note that registers SEQ[n].REFRESH and SEQ[n].ENDDELAY are ignored when DECODER.MODE=NextStep. The next value is loaded upon every received NEXTSTEP task.

SEQ[n].PTR is the pointer used to fetch COMPARE values from RAM. If the SEQ[n].PTR is not pointing to a RAM region, an EasyDMA transfer may result in a HardFault or RAM corruption. See Memory on page 20 for more information about the different memory regions. After the SEQ[n].PTR is set to the desired RAM location, the SEQ[n].CNT register must be set to number of 16-bit half words in the sequence. It is important to observe that the Grouped mode requires one half word per group, while the Single mode requires one half word per channel, thus increasing the RAM size occupation. If PWM generation is not running when the SEQSTART[n] task is triggered, the task will load the first value from RAM and then start the PWM generation. A SEQSTARTED[n] event is generated as soon as the EasyDMA has read the first PWM parameter from RAM and the wave counter has started executing it. When LOOP.CNT=0, sequence n=0 or 1 is played back once. After the last value in the sequence has been loaded and started executing, a SEQEND[n] event is generated. The PWM generation will then continue with the last loaded value. The following figure illustrates an example of such simple playback:

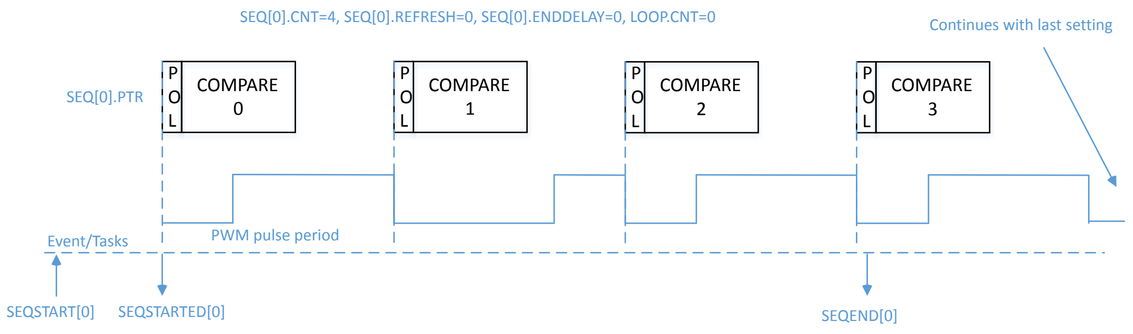


Figure 43: Simple sequence example

Figure depicts the source code used for configuration and timing details in a sequence where only sequence 0 is used and only run once with a new PWM duty cycle for each period.

```
NRF_PWM0->PSEL.OUT[0] = (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                                 PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_Up << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                                 PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
NRF_PWM0->LOOP        = (PWM_LOOP_CNT_Disabled << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER   = (PWM_DECODER_LOAD_Common << PWM_DECODER_LOAD_Pos) |
                      (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
NRF_PWM0->SEQ[0].PTR  = ((uint32_t)(seq0_ram) << PWM_SEQ_PTR_PTR_Pos);
NRF_PWM0->SEQ[0].CNT  = ((sizeof(seq0_ram) / sizeof(uint16_t)) <<
                                                 PWM_SEQ_CNT_CNT_Pos);
NRF_PWM0->SEQ[0].REFRESH  = 0;
NRF_PWM0->SEQ[0].ENDDELAY = 0;
NRF_PWM0->TASKS_SEQSTART[0] = 1;
```

To completely stop the PWM generation and force the associated pins to a defined state, a STOP task can be triggered at any time. A STOPPED event is generated when the PWM generation has stopped at the end of currently running PWM period, and the pins go into their idle state as defined in GPIO OUT register. PWM generation can then only be restarted through a SEQSTART[n] task. SEQSTART[n] will resume PWM generation after having loaded the first value from the RAM buffer defined in the SEQ[n].PTR register.

The table below indicates when specific registers get sampled by the hardware. Care should be taken when updating these registers to avoid that values are applied earlier than expected.

Register	Taken into account by hardware	Recommended (safe) update
SEQ[n].PTR	When sending the SEQSTART[n] task	After having received the SEQSTARTED[n] event
SEQ[n].CNT	When sending the SEQSTART[n] task	After having received the SEQSTARTED[n] event
SEQ[0].ENDDELAY	When sending the SEQSTART[0] task Every time a new value from sequence [0] has been loaded from RAM and gets applied to the Wave Counter (indicated by the PWMPERIODEND event)	Before starting sequence [0] through a SEQSTART[0] task When no more value from sequence [0] gets loaded from RAM (indicated by the SEQEND[0] event) At any time during sequence [1] (which starts when the SEQSTARTED[1] event is generated)
SEQ[1].ENDDELAY	When sending the SEQSTART[1] task Every time a new value from sequence [1] has been loaded from RAM and gets applied to the Wave Counter (indicated by the PWMPERIODEND event)	Before starting sequence [1] through a SEQSTART[1] task When no more value from sequence [1] gets loaded from RAM (indicated by the SEQEND[1] event) At any time during sequence [0] (which starts when the SEQSTARTED[0] event is generated)
SEQ[0].REFRESH	When sending the SEQSTART[0] task Every time a new value from sequence [0] has been loaded from RAM and gets applied to the Wave Counter (indicated by the PWMPERIODEND event)	Before starting sequence [0] through a SEQSTART[0] task At any time during sequence [1] (which starts when the SEQSTARTED[1] event is generated)
SEQ[1].REFRESH	When sending the SEQSTART[1] task Every time a new value from sequence [1] has been loaded from RAM and gets applied to the Wave Counter (indicated by the PWMPERIODEND event)	Before starting sequence [1] through a SEQSTART[1] task At any time during sequence [0] (which starts when the SEQSTARTED[0] event is generated)
COUNTERTOP	In DECODER.LOAD=WaveForm: this register is ignored. In all other LOAD modes: at the end of current PWM period (indicated by the PWMPERIODEND event)	Before starting PWM generation through a SEQSTART[n] task After a STOP task has been triggered, and the STOPPED event has been received.
MODE	Immediately	Before starting PWM generation through a SEQSTART[n] task After a STOP task has been triggered, and the STOPPED event has been received.
DECODER	Immediately	Before starting PWM generation through a SEQSTART[n] task After a STOP task has been triggered, and the STOPPED event has been received.
PRESCALER	Immediately	Before starting PWM generation through a SEQSTART[n] task After a STOP task has been triggered, and the STOPPED event has been received.
LOOP	Immediately	Before starting PWM generation through a SEQSTART[n] task After a STOP task has been triggered, and the STOPPED event has been received.
PSEL.OUT[n]	Immediately	Before enabling the PWM instance through the ENABLE register

Table 55: When to safely update PWM registers

Note: SEQ[n].REFRESH and SEQ[n].ENDDELAY are ignored at the end of a complex sequence, indicated by a LOOPSDONE event. The reason for this is that the last value loaded from RAM is maintained until further action from software (restarting a new sequence, or stopping PWM generation).

A more complex example, where LOOP.CNT>0, is shown in the following figure:

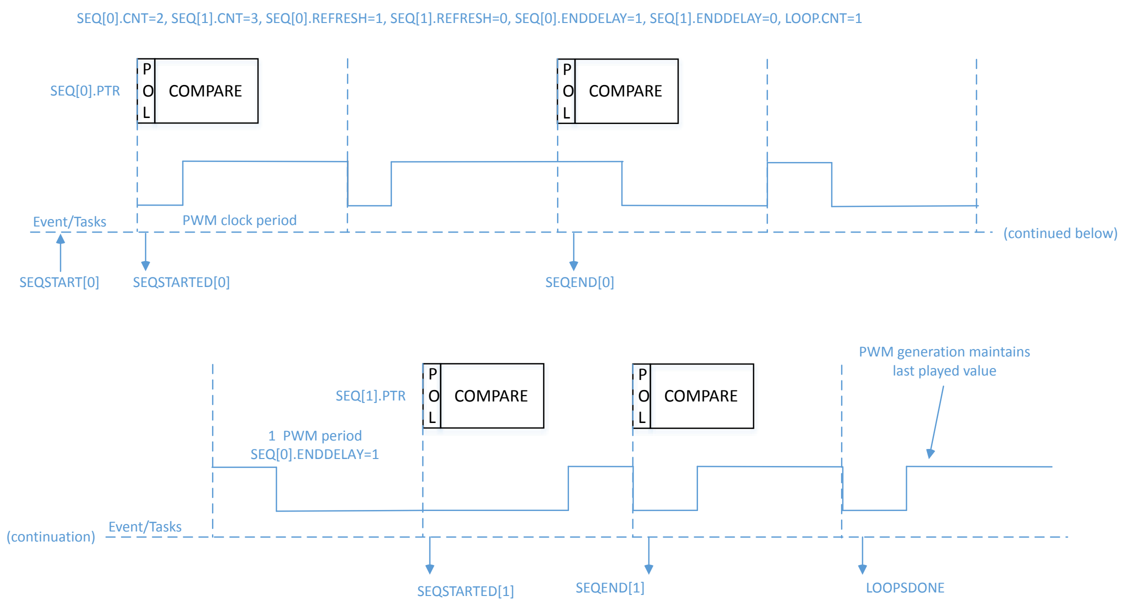


Figure 44: Example using two sequences

In this case, an automated playback takes place, consisting of SEQ[0], delay 0, SEQ[1], delay 1, then again SEQ[0], etc. The user can choose to start a complex playback with SEQ[0] or SEQ[1] through sending the SEQSTART[0] or SEQSTART[1] task. The complex playback always ends with delay 1.

The two sequences 0 and 1 are defined by the addresses of value tables in RAM (pointed to by SEQ[n].PTR) and the buffer size (SEQ[n].CNT). The rate at which a new value is loaded is defined individually for each sequence by SEQ[n].REFRESH. The chaining of sequence 1 following the sequence 0 is implicit, the LOOP.CNT register allows the chaining of sequence 1 to sequence 0 for a determined number of times. In other words, it allows to repeat a complex sequence a number of times in a fully automated way.

In the following code example, sequence 0 is defined with SEQ[0].REFRESH set to 1, meaning that a new PWM duty cycle is pushed every second PWM period. This complex sequence is started with the SEQSTART[0] task, so SEQ[0] is played first. Since SEQ[0].ENDDELAY=1 there will be one PWM period delay between last period on sequence 0 and the first period on sequence 1. Since SEQ[1].ENDDELAY=0 there is no delay 1, so SEQ[0] would be started immediately after the end of SEQ[1]. However, as LOOP.CNT is

1, the playback stops after having played SEQ[1] only once, and both SEQEND[1] and LOOPSDONE are generated (their order is not guaranteed in this case).

```
NRF_PWM0->PSEL.OUT[0] = (first_pin << PWM_PSEL_OUT_PIN_Pos) |
                        (PWM_PSEL_OUT_CONNECT_Connected <<
                                                 PWM_PSEL_OUT_CONNECT_Pos);
NRF_PWM0->ENABLE      = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
NRF_PWM0->MODE        = (PWM_MODE_UPDOWN_Up << PWM_MODE_UPDOWN_Pos);
NRF_PWM0->PRESCALER   = (PWM_PRESCALER_PRESCALER_DIV_1 <<
                                                 PWM_PRESCALER_PRESCALER_Pos);
NRF_PWM0->COUNTERTOP  = (16000 << PWM_COUNTERTOP_COUNTERTOP_Pos); //1 msec
NRF_PWM0->LOOP        = (1 << PWM_LOOP_CNT_Pos);
NRF_PWM0->DECODER   = (PWM_DECODER_LOAD_Common << PWM_DECODER_LOAD_Pos) |
                      (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
NRF_PWM0->SEQ[0].PTR  = ((uint32_t)(seq0_ram) << PWM_SEQ_PTR_PTR_Pos);
NRF_PWM0->SEQ[0].CNT  = ((sizeof(seq0_ram) / sizeof(uint16_t)) <<
                                                 PWM_SEQ_CNT_CNT_Pos);
NRF_PWM0->SEQ[0].REFRESH  = 1;
NRF_PWM0->SEQ[0].ENDDELAY = 1;
NRF_PWM0->SEQ[1].PTR  = ((uint32_t)(seq1_ram) << PWM_SEQ_PTR_PTR_Pos);
NRF_PWM0->SEQ[1].CNT  = ((sizeof(seq1_ram) / sizeof(uint16_t)) <<
                                                 PWM_SEQ_CNT_CNT_Pos);
NRF_PWM0->SEQ[1].REFRESH  = 0;
NRF_PWM0->SEQ[1].ENDDELAY = 0;
NRF_PWM0->TASKS_SEQSTART[0] = 1;
```

The decoder can also be configured to asynchronously load new PWM duty cycle. If the DECODER.MODE register is set to NextStep, then the NEXTSTEP task will cause an update of internal compare registers on the next PWM period.

The following figures provide an overview of each part of an arbitrary sequence, in various modes (LOOP.CNT=0 and LOOP.CNT>0). In particular, the following are represented:

- Initial and final duty cycle on the PWM output(s)
- Chaining of SEQ[0] and SEQ[1] if LOOP.CNT>0
- Influence of registers on the sequence
- Events generated during a sequence
- DMA activity (loading of next value and applying it to the output(s))

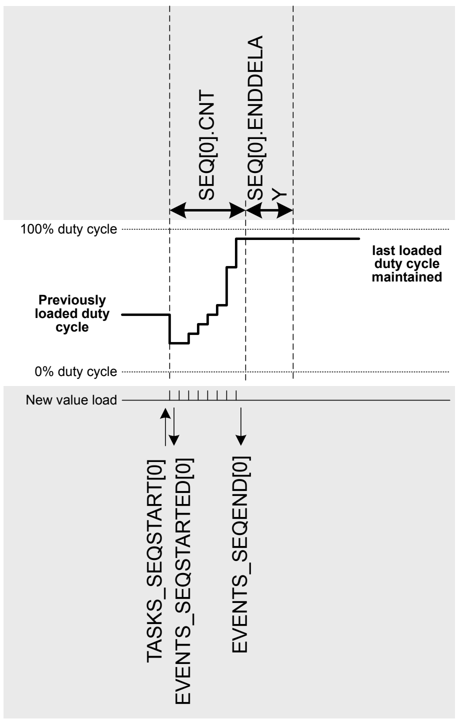


Figure 45: Single shot (LOOP.CNT=0)

Note: The single-shot example also applies to SEQ[1]. Only SEQ[0] is represented for simplicity.

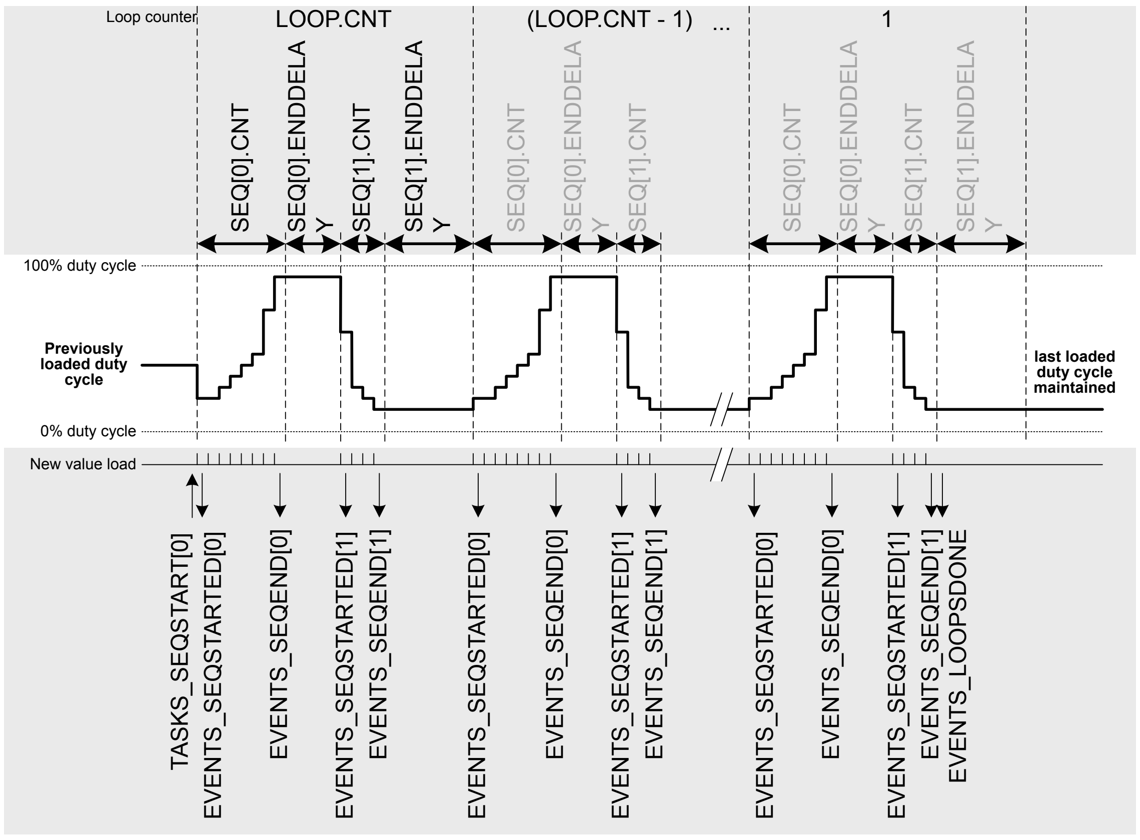


Figure 46: Complex sequence (LOOP.CNT>0) starting with SEQ[0]

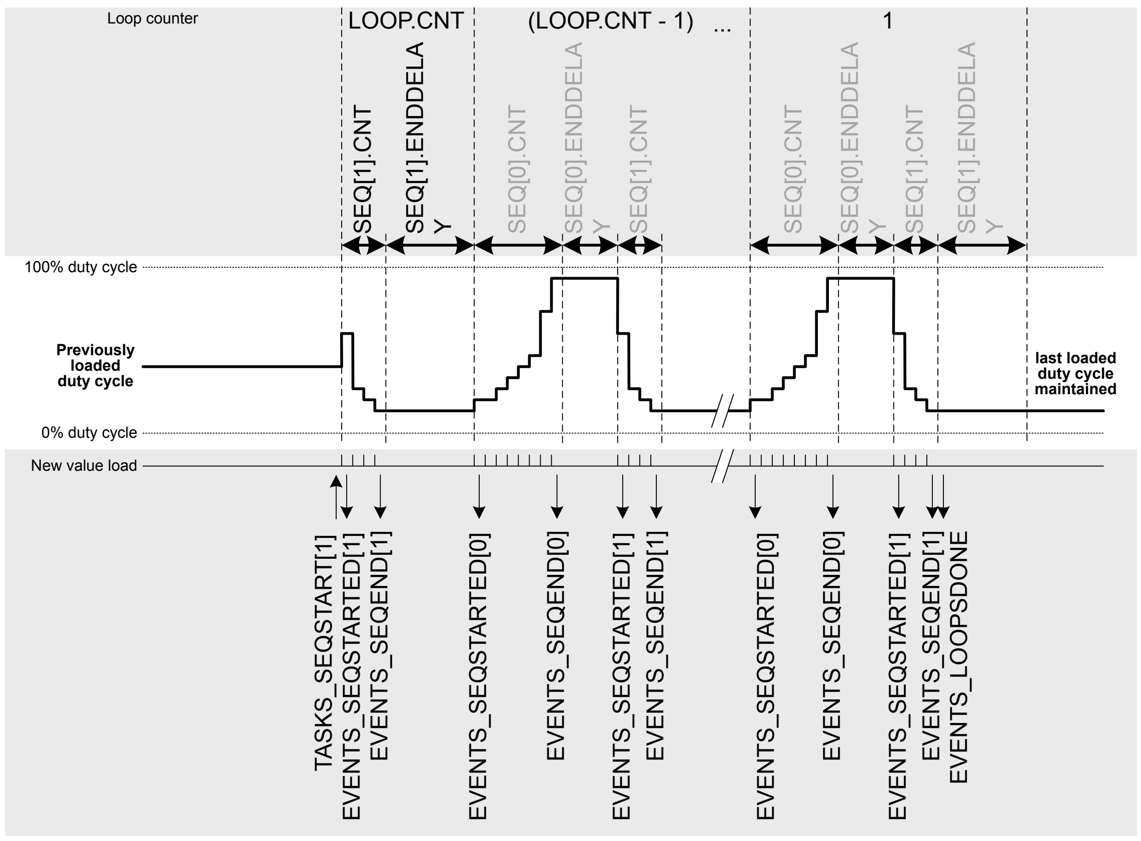


Figure 47: Complex sequence (LOOP.CNT>0) starting with SEQ[1]

Note: If a sequence is in use in a simple or complex sequence, it must have a length of SEQ[n].CNT > 0.

6.12.3 Limitations

Previous compare value is repeated if the PWM period is shorter than the time it takes for the EasyDMA to retrieve from RAM and update the internal compare registers. This is to ensure a glitch-free operation even for very short PWM periods.

6.12.4 Pin configuration

The OUT[n] (n=0..3) signals associated with each PWM channel are mapped to physical pins according to the configuration of PSEL.OUT[n] registers. If PSEL.OUT[n].CONNECT is set to Disconnected, the associated PWM module signal will not be connected to any physical pins.

The PSEL.OUT[n] registers and their configurations are used as long as the PWM module is enabled and the PWM generation active (wave counter started). They are retained only as long as the device is in System ON mode (see section POWER for more information about power modes).

To ensure correct behavior in the PWM module, the pins that are used must be configured in the GPIO peripheral in the following way before the PWM module is enabled:

PWM signal	PWM pin	Direction	Output value	Comment
OUT[n]	As specified in PSEL.OUT[n] (n=0..3)	Output	0	Idle state defined in GPIO OUT register

Table 56: Recommended GPIO configuration before starting PWM generation

The idle state of a pin is defined by the OUT register in the GPIO module, to ensure that the pins used by the PWM module are driven correctly. If PWM generation is stopped by triggering a STOP task, the PWM module itself is temporarily disabled or the device temporarily enters System OFF. This configuration must be retained in the GPIO for the selected pins (I/Os) for as long as the PWM module is supposed to be connected to an external PWM circuit.

Only one peripheral can be assigned to drive a particular GPIO pin at a time. Failing to do so may result in unpredictable behavior.

6.12.5 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50021000 0x40021000	PWM	PWM0 : S PWM0 : NS	US	SA	Pulse width modulation unit 0	
0x50022000 0x40022000	PWM	PWM1 : S PWM1 : NS	US	SA	Pulse width modulation unit 1	
0x50023000 0x40023000	PWM	PWM2 : S PWM2 : NS	US	SA	Pulse width modulation unit 2	
0x50024000 0x40024000	PWM	PWM3 : S PWM3 : NS	US	SA	Pulse width modulation unit 3	

Table 57: Instances

Register	Offset	Security	Description
TASKS_STOP	0x004		Stops PWM pulse generation on all channels at the end of current PWM period, and stops sequence playback
TASKS_SEQSTART[0]	0x008		Loads the first PWM value on all enabled channels from sequence 0, and starts playing that sequence at the rate defined in SEQ[0]REFRESH and/or DECODER.MODE. Causes PWM generation to start if not running.
TASKS_SEQSTART[1]	0x00C		Loads the first PWM value on all enabled channels from sequence 1, and starts playing that sequence at the rate defined in SEQ[1]REFRESH and/or DECODER.MODE. Causes PWM generation to start if not running.
TASKS_NEXTSTEP	0x010		Steps by one value in the current sequence on all enabled channels if DECODER.MODE=NextStep. Does not cause PWM generation to start if not running.
SUBSCRIBE_STOP	0x084		Subscribe configuration for task STOP
SUBSCRIBE_SEQSTART[0]	0x088		Subscribe configuration for task SEQSTART[0]
SUBSCRIBE_SEQSTART[1]	0x08C		Subscribe configuration for task SEQSTART[1]
SUBSCRIBE_NEXTSTEP	0x090		Subscribe configuration for task NEXTSTEP
EVENTS_STOPPED	0x104		Response to STOP task, emitted when PWM pulses are no longer generated
EVENTS_SEQSTARTED[0]	0x108		First PWM period started on sequence 0
EVENTS_SEQSTARTED[1]	0x10C		First PWM period started on sequence 1
EVENTS_SEQEND[0]	0x110		Emitted at end of every sequence 0, when last value from RAM has been applied to wave counter
EVENTS_SEQEND[1]	0x114		Emitted at end of every sequence 1, when last value from RAM has been applied to wave counter
EVENTS_PWMPERIODEND	0x118		Emitted at the end of each PWM period
EVENTS_LOOPSDONE	0x11C		Concatenated sequences have been played the amount of times defined in LOOP.CNT
PUBLISH_STOPPED	0x184		Publish configuration for event STOPPED
PUBLISH_SEQSTARTED[0]	0x188		Publish configuration for event SEQSTARTED[0]
PUBLISH_SEQSTARTED[1]	0x18C		Publish configuration for event SEQSTARTED[1]
PUBLISH_SEQEND[0]	0x190		Publish configuration for event SEQEND[0]
PUBLISH_SEQEND[1]	0x194		Publish configuration for event SEQEND[1]

Register	Offset	Security	Description
PUBLISH_PWMPERIODEND	0x198		Publish configuration for event PWMPERIODEND
PUBLISH_LOOPSDONE	0x19C		Publish configuration for event LOOPSDONE
SHORTS	0x200		Shortcuts between local events and tasks
INTEN	0x300		Enable or disable interrupt
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
ENABLE	0x500		PWM module enable register
MODE	0x504		Selects operating mode of the wave counter
COUNTERTOP	0x508		Value up to which the pulse generator counter counts
PRESCALER	0x50C		Configuration for PWM_CLK
DECODER	0x510		Configuration of the decoder
LOOP	0x514		Number of playbacks of a loop
SEQ[0].PTR	0x520		Beginning address in RAM of this sequence
SEQ[0].CNT	0x524		Number of values (duty cycles) in this sequence
SEQ[0].REFRESH	0x528		Number of additional PWM periods between samples loaded into compare register
SEQ[0].ENDDELAY	0x52C		Time added after the sequence
SEQ[1].PTR	0x540		Beginning address in RAM of this sequence
SEQ[1].CNT	0x544		Number of values (duty cycles) in this sequence
SEQ[1].REFRESH	0x548		Number of additional PWM periods between samples loaded into compare register
SEQ[1].ENDDELAY	0x54C		Time added after the sequence
PSEL.OUT[0]	0x560		Output pin select for PWM channel 0
PSEL.OUT[1]	0x564		Output pin select for PWM channel 1
PSEL.OUT[2]	0x568		Output pin select for PWM channel 2
PSEL.OUT[3]	0x56C		Output pin select for PWM channel 3

Table 58: Register overview

6.12.5.1 TASKS_STOP

Address offset: 0x004

Stops PWM pulse generation on all channels at the end of current PWM period, and stops sequence playback

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_STOP			Stops PWM pulse generation on all channels at the end of current PWM period, and stops sequence playback
			Trigger	1	Trigger task

6.12.5.2 TASKS_SEQSTART[n] (n=0..1)

Address offset: 0x008 + (n × 0x4)

Loads the first PWM value on all enabled channels from sequence n, and starts playing that sequence at the rate defined in SEQ[n]REFRESH and/or DECODER.MODE. Causes PWM generation to start if not running.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_SEQSTART			Loads the first PWM value on all enabled channels from sequence n, and starts playing that sequence at the rate defined in SEQ[n]REFRESH and/or DECODER.MODE. Causes PWM generation to start if not running.
			Trigger	1	Trigger task

6.12.5.3 TASKS_NEXTSTEP

Address offset: 0x010

Steps by one value in the current sequence on all enabled channels if DECODER.MODE=NextStep. Does not cause PWM generation to start if not running.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_NEXTSTEP			Steps by one value in the current sequence on all enabled channels if DECODER.MODE=NextStep. Does not cause PWM generation to start if not running.
			Trigger	1	Trigger task

6.12.5.4 SUBSCRIBE_STOP

Address offset: 0x084

Subscribe configuration for task STOP

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task STOP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.12.5.5 SUBSCRIBE_SEQSTART[n] (n=0..1)

Address offset: 0x088 + (n × 0x4)

Subscribe configuration for task SEQSTART[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task SEQSTART[n] will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.12.5.6 SUBSCRIBE_NEXTSTEP

Address offset: 0x090

Subscribe configuration for task NEXTSTEP

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task NEXTSTEP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.12.5.7 EVENTS_STOPPED

Address offset: 0x104

Response to STOP task, emitted when PWM pulses are no longer generated

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_STOPPED			Response to STOP task, emitted when PWM pulses are no longer generated
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.12.5.8 EVENTS_SEQSTARTED[n] (n=0..1)

Address offset: 0x108 + (n × 0x4)

First PWM period started on sequence n

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_SEQSTARTED			First PWM period started on sequence n
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.12.5.9 EVENTS_SEQEND[n] (n=0..1)

Address offset: 0x110 + (n × 0x4)

Emitted at end of every sequence n, when last value from RAM has been applied to wave counter

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_SEQEND			Emitted at end of every sequence n, when last value from RAM has been applied to wave counter
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.12.5.10 EVENTS_PWMPERIODEND

Address offset: 0x118

Emitted at the end of each PWM period

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_PWMPERIODEND			Emitted at the end of each PWM period
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.12.5.11 EVENTS_LOOPSDONE

Address offset: 0x11C

Concatenated sequences have been played the amount of times defined in LOOP.CNT

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_LOOPSDONE			Concatenated sequences have been played the amount of times defined in LOOP.CNT
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.12.5.12 PUBLISH_STOPPED

Address offset: 0x184

Publish configuration for event STOPPED

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event STOPPED will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.12.5.13 PUBLISH_SEQSTARTED[n] (n=0..1)

Address offset: 0x188 + (n × 0x4)

Publish configuration for event SEQSTARTED[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event SEQSTARTED[n] will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.12.5.14 PUBLISH_SEQEND[n] (n=0..1)

Address offset: 0x190 + (n × 0x4)

Publish configuration for event SEQEND[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event SEQEND[n] will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.12.5.15 PUBLISH_PWMPERIODEND

Address offset: 0x198

Publish configuration for event PWMPERIODEND

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event PWMPERIODEND will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.12.5.16 PUBLISH_LOOPSDONE

Address offset: 0x19C

Publish configuration for event LOOPSDONE

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event LOOPSDONE will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.12.5.17 SHORTS

Address offset: 0x200

Shortcuts between local events and tasks

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																												E	D	C	B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	SEQEND0_STOP			Shortcut between event SEQEND[0] and task STOP
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
B	RW	SEQEND1_STOP			Shortcut between event SEQEND[1] and task STOP
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
C	RW	LOOPSDONE_SEQSTART0			Shortcut between event LOOPSDONE and task SEQSTART[0]
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
D	RW	LOOPSDONE_SEQSTART1			Shortcut between event LOOPSDONE and task SEQSTART[1]
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut
E	RW	LOOPSDONE_STOP			Shortcut between event LOOPSDONE and task STOP
			Disabled	0	Disable shortcut
			Enabled	1	Enable shortcut

6.12.5.18 INTEN

Address offset: 0x300

Enable or disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
B	RW	STOPPED			Enable or disable interrupt for event STOPPED
			Disabled	0	Disable
			Enabled	1	Enable
C-D	RW	SEQSTARTED[i] (i=0..1)			Enable or disable interrupt for event SEQSTARTED[i]
			Disabled	0	Disable
			Enabled	1	Enable
E-F	RW	SEQEND[i] (i=0..1)			Enable or disable interrupt for event SEQEND[i]
			Disabled	0	Disable
			Enabled	1	Enable
G	RW	PWMPERIODEND			Enable or disable interrupt for event PWMPERIODEND

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
			Disabled	0	Disable
			Enabled	1	Enable
H	RW	LOOPSDONE			Enable or disable interrupt for event LOOPSDONE
			Disabled	0	Disable
			Enabled	1	Enable

6.12.5.19 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
B	RW	STOPPED			Write '1' to enable interrupt for event STOPPED
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C-D	RW	SEQSTARTED[i] (i=0..1)			Write '1' to enable interrupt for event SEQSTARTED[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
E-F	RW	SEQEND[i] (i=0..1)			Write '1' to enable interrupt for event SEQEND[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
G	RW	PWMPERIODEND			Write '1' to enable interrupt for event PWMPERIODEND
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
H	RW	LOOPSDONE			Write '1' to enable interrupt for event LOOPSDONE
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.12.5.20 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
B	RW	STOPPED			Write '1' to disable interrupt for event STOPPED
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																									H	G	F	E	D	C	B	
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
C-D	RW	SEQSTARTED[i] (i=0..1)			Write '1' to disable interrupt for event SEQSTARTED[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
E-F	RW	SEQEND[i] (i=0..1)			Write '1' to disable interrupt for event SEQEND[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
G	RW	PWMPERIODEND			Write '1' to disable interrupt for event PWMPERIODEND
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
H	RW	LOOPSDONE			Write '1' to disable interrupt for event LOOPSDONE
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.12.5.21 ENABLE

Address offset: 0x500

PWM module enable register

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	ENABLE			Enable or disable PWM module
			Disabled	0	Disabled
			Enabled	1	Enable

6.12.5.22 MODE

Address offset: 0x504

Selects operating mode of the wave counter

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	UPDOWN			Selects up mode or up-and-down mode for the counter
			Up	0	Up counter, edge-aligned PWM duty cycle
			UpAndDown	1	Up and down counter, center-aligned PWM duty cycle

6.12.5.23 COUNTERTOP

Address offset: 0x508

Value up to which the pulse generator counter counts

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																		A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x000003FF	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	COUNTERTOP		[3..32767]	Value up to which the pulse generator counter counts. This register is ignored when DECODER.MODE=WaveForm and only values from RAM are used.

6.12.5.24 PRESCALER

Address offset: 0x50C

Configuration for PWM_CLK

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																														A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	PRESCALER			Prescaler of PWM_CLK
			DIV_1	0	Divide by 1 (16 MHz)
			DIV_2	1	Divide by 2 (8 MHz)
			DIV_4	2	Divide by 4 (4 MHz)
			DIV_8	3	Divide by 8 (2 MHz)
			DIV_16	4	Divide by 16 (1 MHz)
			DIV_32	5	Divide by 32 (500 kHz)
			DIV_64	6	Divide by 64 (250 kHz)
			DIV_128	7	Divide by 128 (125 kHz)

6.12.5.25 DECODER

Address offset: 0x510

Configuration of the decoder

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																								B							A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	LOAD			How a sequence is read from RAM and spread to the compare register
			Common	0	1st half word (16-bit) used in all PWM channels 0..3
			Grouped	1	1st half word (16-bit) used in channel 0..1; 2nd word in channel 2..3
			Individual	2	1st half word (16-bit) in ch.0; 2nd in ch.1; ...; 4th in ch.3
			WaveForm	3	1st half word (16-bit) in ch.0; 2nd in ch.1; ...; 4th in COUNTERTOP
B	RW	MODE			Selects source for advancing the active sequence
			RefreshCount	0	SEQ[n].REFRESH is used to determine loading internal compare registers
			NextStep	1	NEXTSTEP task causes a new value to be loaded to internal compare registers

6.12.5.26 LOOP

Address offset: 0x514

Number of playbacks of a loop

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CNT			Number of playbacks of pattern cycles
			Disabled	0	Looping disabled (stop at the end of the sequence)

6.12.5.27 SEQ[n].PTR (n=0..1)

Address offset: 0x520 + (n × 0x20)

Beginning address in RAM of this sequence

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	PTR			Beginning address in RAM of this sequence

Note: See the memory chapter for details about which memories are available for EasyDMA.

6.12.5.28 SEQ[n].CNT (n=0..1)

Address offset: 0x524 + (n × 0x20)

Number of values (duty cycles) in this sequence

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																		A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CNT			Number of values (duty cycles) in this sequence
			Disabled	0	Sequence is disabled, and shall not be started as it is empty

6.12.5.29 SEQ[n].REFRESH (n=0..1)

Address offset: 0x528 + (n × 0x20)

Number of additional PWM periods between samples loaded into compare register

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID									A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000001	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	CNT			Number of additional PWM periods between samples loaded into compare register (load every REFRESH.CNT+1 PWM periods)
			Continuous	0	Update every PWM period

6.12.5.30 SEQ[n].ENDDELAY (n=0..1)

Address offset: 0x52C + (n × 0x20)

Time added after the sequence

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID									A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CNT			Time added after the sequence in PWM periods

6.12.5.31 PSEL.OUT[n] (n=0..3)

Address offset: 0x560 + (n × 0x4)

Output pin select for PWM channel n

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	C																											A	A	A	A	A
Reset 0xFFFFFFFF	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ID	Acce	Field	Value ID	Value	Description
A	RW	PIN		[0..31]	Pin number
C	RW	CONNECT			Connection
			Disconnected	1	Disconnect
			Connected	0	Connect

6.13 RTC — Real-time counter

The real-time counter (RTC) module provides a generic, low power timer on the low frequency clock source (LFCLK).

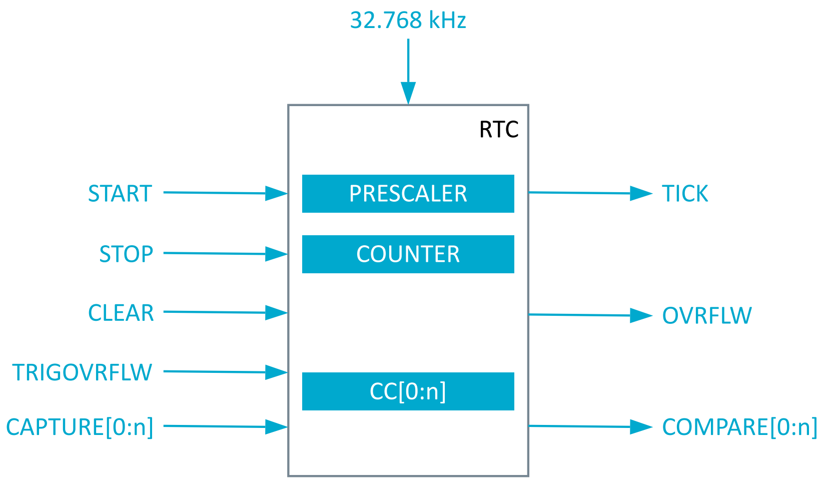


Figure 48: RTC block diagram

The RTC module features a 24-bit COUNTER, a 12-bit (1/X) prescaler, capture/compare registers, and a tick event generator for low power, tickless RTOS implementation.

6.13.1 Clock source

The RTC will run off the LFCLK.

When started, the RTC will automatically request the LFCLK source with RC oscillator if the LFCLK is not already running.

See CLOCK — Clock control on page 64 for more information about clock sources.

6.13.2 Resolution versus overflow and the prescaler

The relationship between the prescaler, counter resolution and overflow is summarized in a table.

Prescaler	Counter resolution	Overflow
0	30.517 µs	512 seconds
2^8-1	7812.5 µs	131072 seconds
2^{12}-1	125 ms	582.542 hours

Table 59: RTC resolution versus overflow

Counter increment frequency is given by the following equation:

```
fRTC [kHz] = 32.768 / (PRESCALER + 1 )
```

The PRESCALER register is read/write when the RTC is stopped. Once the RTC is started, the prescaler register is read-only and thus writing to it when the RTC is started has no effect.

The prescaler is restarted on tasks START, CLEAR and TRIGOVRFLW. That is, the prescaler value is latched to an internal register (<<PRESC>>) on these tasks.

Examples:

1. Desired COUNTER frequency 100 Hz (10 ms counter period)

   PRESCALER = round(32.768 kHz / 100 Hz) - 1 = 327

   f_{RTC} = 99.9 Hz

   10009.576 µs counter period
2. Desired COUNTER frequency 8 Hz (125 ms counter period)

   PRESCALER = round(32.768 kHz / 8 Hz) – 1 = 4095

   f_{RTC} = 8 Hz

   125 ms counter period

6.13.3 Counter register

The counter increments on LFCLK when the internal PRESCALER register (<<PRESC>>) is 0x00. <<PRESC>> is reloaded from the PRESCALER register. If enabled, the TICK event occurs on each increment of the COUNTER. The TICK event is disabled by default.

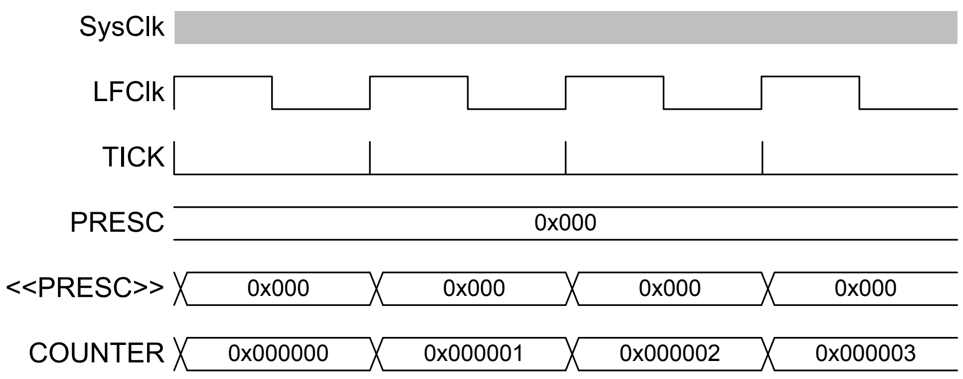


Figure 49: Timing diagram - COUNTER_PRESCALER_0

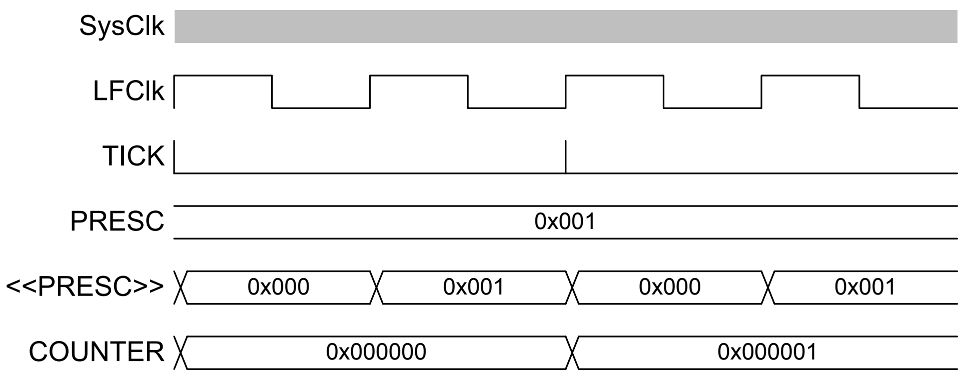


Figure 50: Timing diagram - COUNTER_PRESCALER_1

6.13.4 Overflow

An OVRFLW event is generated on COUNTER register overflow (overflowing from 0xFFFFFF to 0).

The TRIGOVRFLW task will then set the COUNTER value to 0xFFFFF0, to allow software test of the overflow condition.

Note: The OVRFLW event is disabled by default.

6.13.5 Tick event

The TICK event enables low power tick-less RTOS implementation, as it optionally provides a regular interrupt source for an RTOS without the need to use the ARM® SysTick feature.

Using the TICK event, rather than the SysTick, allows the CPU to be powered down while still keeping RTOS scheduling active.

Note: The TICK event is disabled by default.

6.13.6 Event control

To optimize RTC power consumption, events in the RTC can be individually disabled to prevent PCLK16M and HFCLK from being requested when those events are triggered. This is managed using the EVTEN register.

For example, if the TICK event is not required for an application, it should be disabled, since its frequent occurrences may increase power consumption when HFCLK otherwise could be powered down for long periods of time.

This means that the RTC implements a slightly different task and event system compared to the standard system described in Peripheral interface on page 15. The RTC task and event system is illustrated in the figure below.

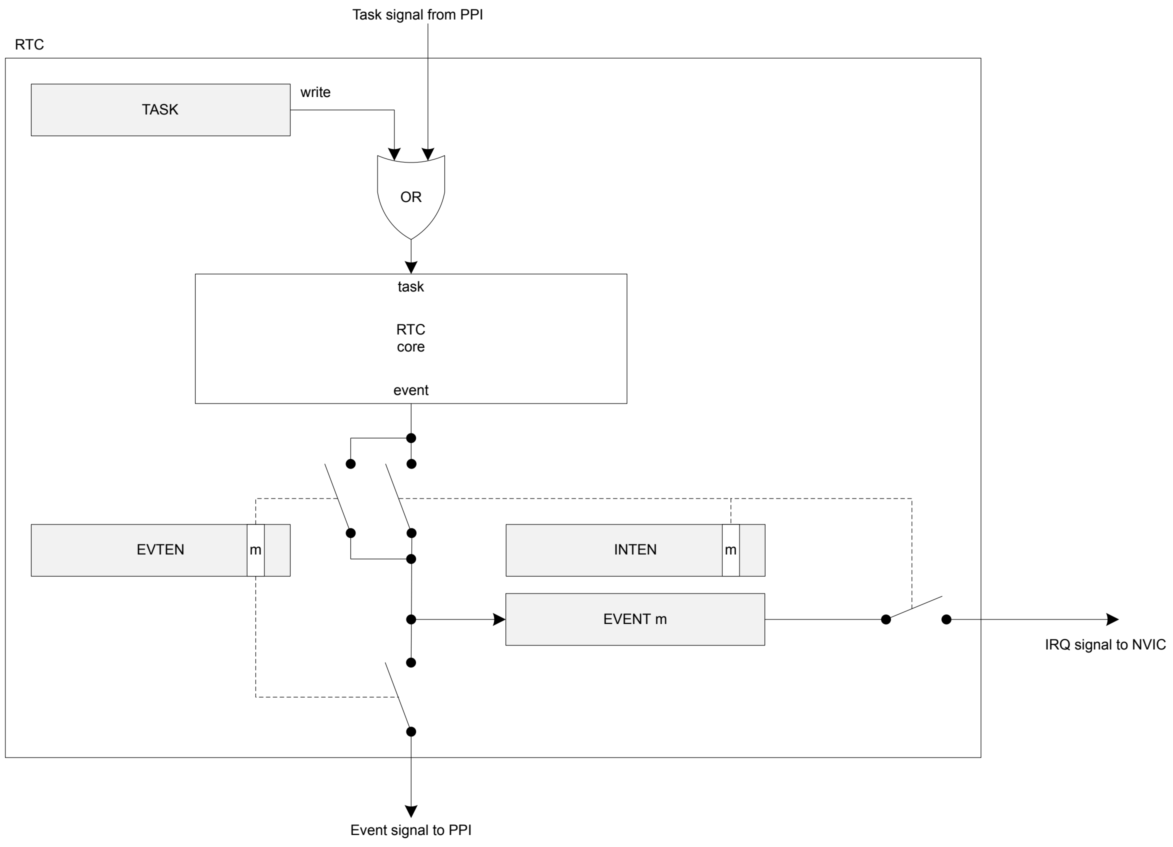


Figure 51: Tasks, events and interrupts in the RTC

6.13.7 Compare

The RTC implements one COMPARE event for every available capture/compare register.

When the COUNTER is incremented and then becomes equal to the value specified in the capture compare register CC[n], the corresponding compare event COMPARE[n] is generated.

When setting a compare register, the following behavior of the RTC COMPARE event should be noted:

- If a CC register value is 0 when a CLEAR task is set, this will not trigger a COMPARE event.

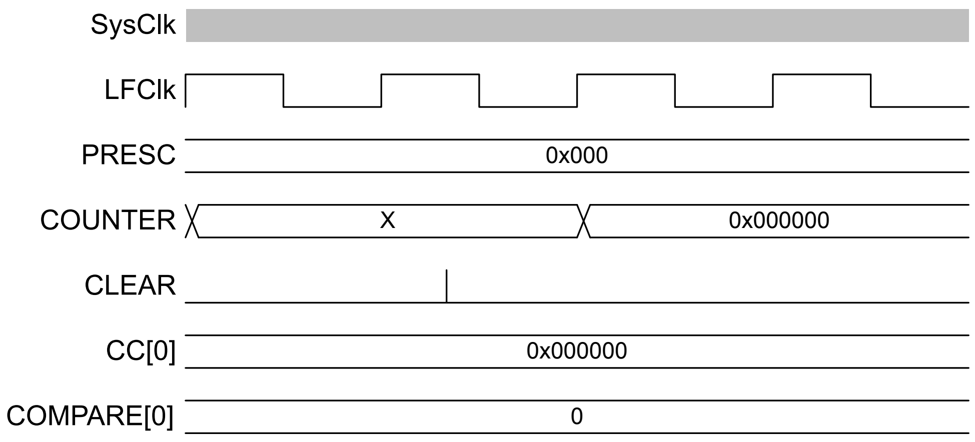


Figure 52: Timing diagram - COMPARE_CLEAR

- If a CC register is N and the COUNTER value is N when the START task is set, this will not trigger a COMPARE event.

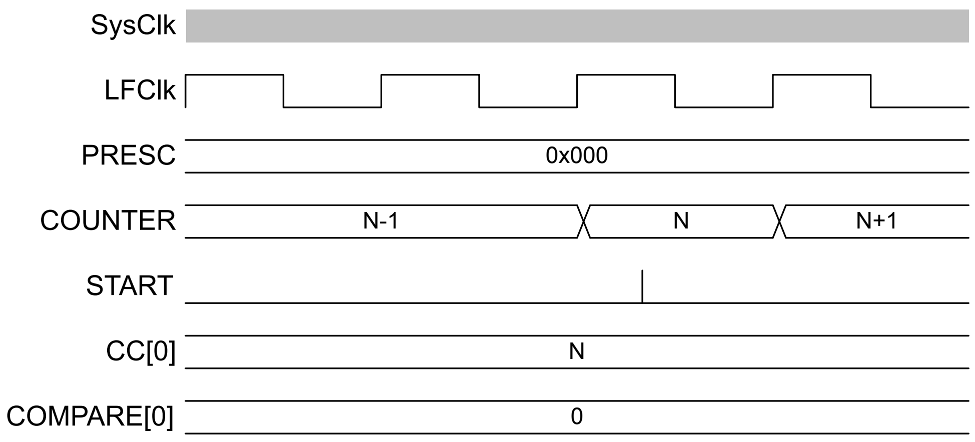


Figure 53: Timing diagram - COMPARE_START

- A COMPARE event occurs when a CC register is N and the COUNTER value transitions from N-1 to N.

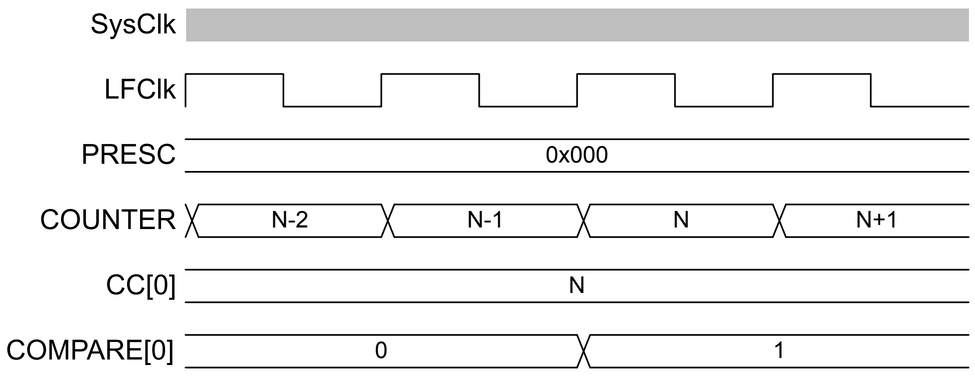


Figure 54: Timing diagram - COMPARE

- If the COUNTER is N, writing N+2 to a CC register is guaranteed to trigger a COMPARE event at N+2.

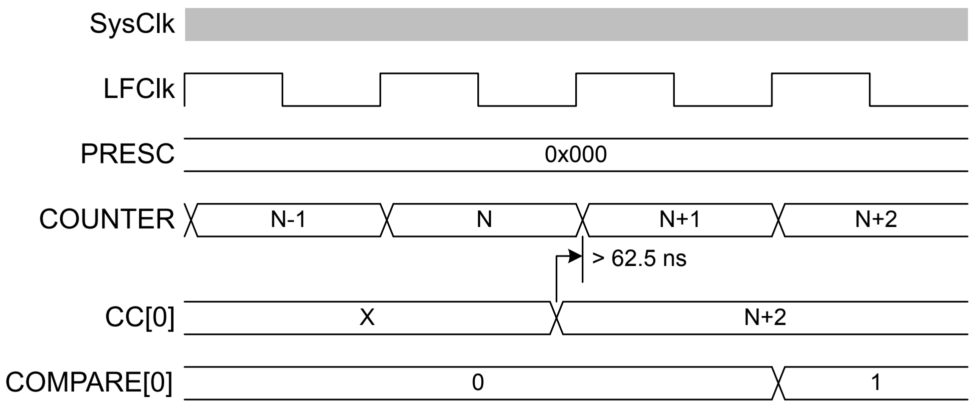


Figure 55: Timing diagram - COMPARE_N+2

- If the COUNTER is N, writing N or N+1 to a CC register may not trigger a COMPARE event.

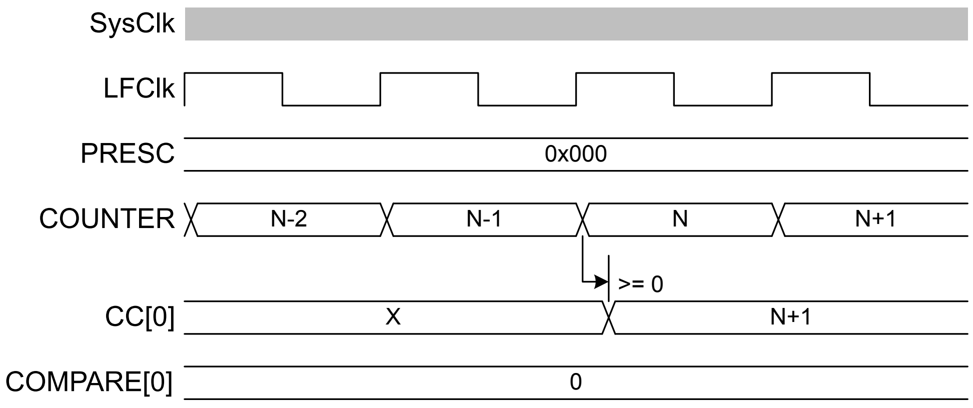


Figure 56: Timing diagram - COMPARE_N+1

- If the COUNTER is N and the current CC register value is N+1 or N+2 when a new CC value is written, a match may trigger on the previous CC value before the new value takes effect. If the current CC value greater than N+2 when the new value is written, there will be no event due to the old value.

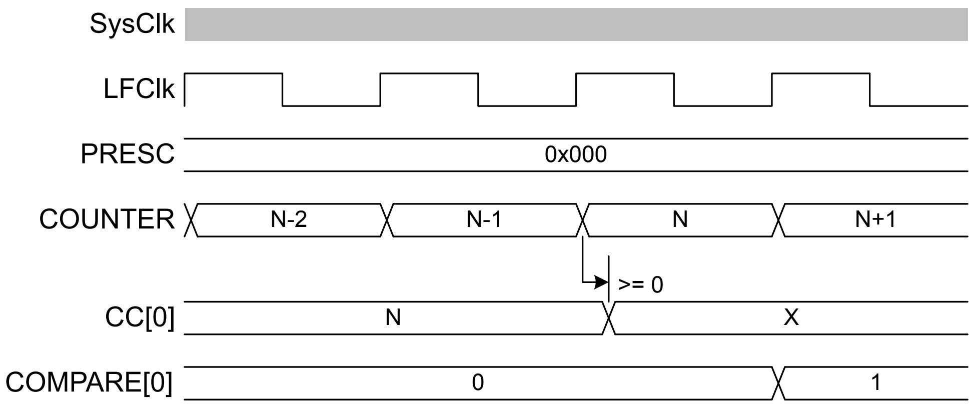


Figure 57: Timing diagram - COMPARE_N-1

6.13.8 Task and event jitter/delay

Jitter or delay in the RTC is due to the peripheral clock being a low frequency clock (LFCLK) which is not synchronous to the faster PCLK16M.

Registers in the peripheral interface, part of the PCLK16M domain, have a set of mirrored registers in the LFCLK domain. For example, the COUNTER value accessible from the CPU is in the PCLK16M domain and is latched on read from an internal COUNTER register in the LFCLK domain. The COUNTER register is modified each time the RTC ticks. The registers are synchronised between the two clock domains (PCLK16M and LFCLK).

1. CLEAR and STOP (and TRIGOVRFLW; not shown) will be delayed as long as it takes for the peripheral to clock a falling edge and rising of the LFCLK. This is between 15.2585 µs and 45.7755 µs – rounded to 15 µs and 46 µs for the remainder of the section.

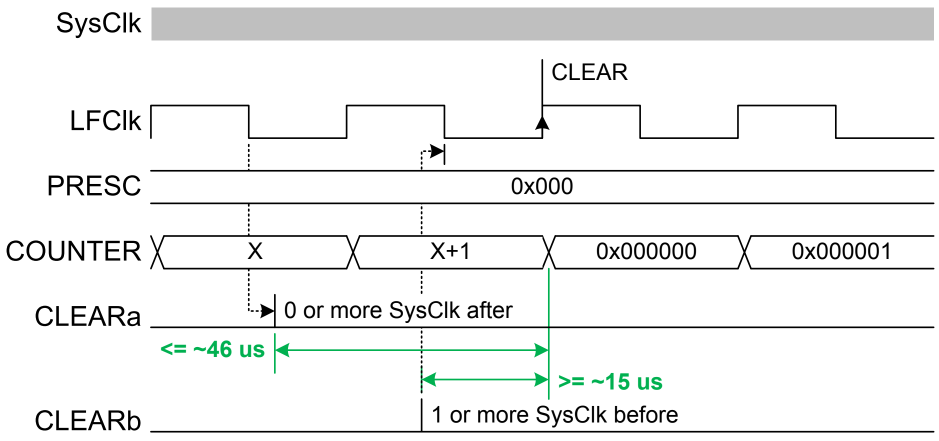


Figure 58: Timing diagram - DELAY_CLEAR

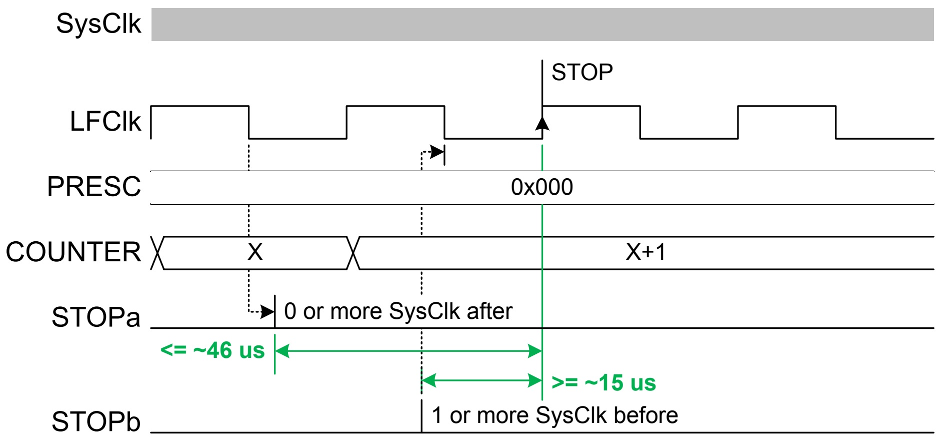


Figure 59: Timing diagram - DELAY_STOP

2. The START task will start the RTC. Assuming that the LFCLK was previously running and stable, the first increment of COUNTER (and instance of TICK event) will be typically after 30.5 µs +/-15 µs. In some cases, in particular if the RTC is started before the LFCLK is running, that timing can be up to ~250 µs. The software should therefore wait for the first TICK if it has to make sure the RTC is running. Sending a TRIGOVRFLW task sets the COUNTER to a value close to overflow. However, since the update of COUNTER relies on a stable LFCLK, sending this task while LFCLK is not running will start LFCLK, but the update will then be delayed by the same amount of time of up to ~250 µs. The figures show the smallest and largest delays on the START task, appearing as a +/-15 µs jitter on the first COUNTER increment.

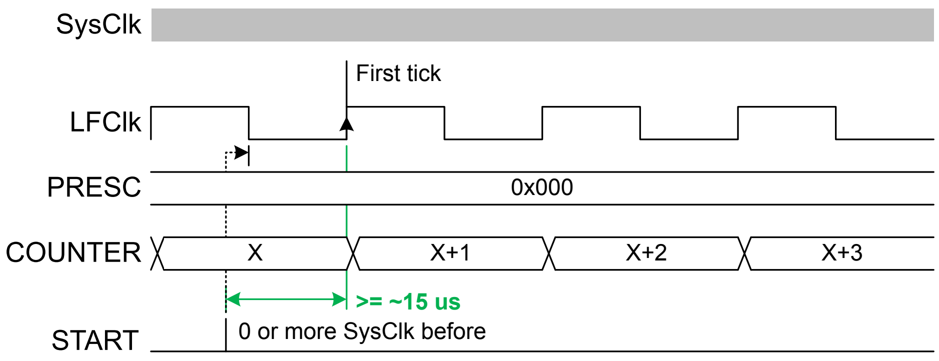


Figure 60: Timing diagram - JITTER_START-

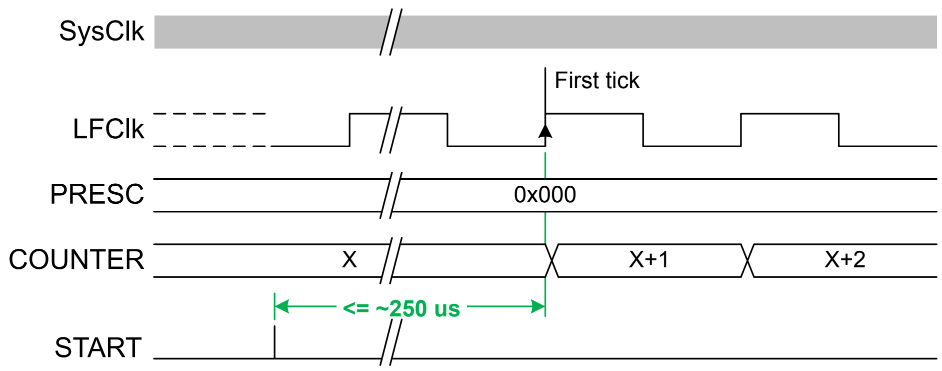


Figure 61: Timing diagram - JITTER_START+

Tables below summarize the jitter introduced on tasks and events.

Task	Delay
CLEAR, START, STOP, TRIGOVRFLOW	+15 to 46 µs

Table 60: RTC jitter magnitudes on tasks

Operation/Function	Jitter
START to COUNTER increment	+/- 15 µs
COMPARE to COMPARE [8]	+/- 62.5 ns

Table 61: RTC jitter magnitudes on events

6.13.9 Reading the counter register

To read the COUNTER register, the internal <<COUNTER>> value is sampled.

To ensure that the <<COUNTER>> is safely sampled (considering that an LFCLK transition may occur during a read), the CPU and core memory bus are halted for three cycles by lowering the core PREADY signal. The read takes the CPU two cycles in addition, resulting in the COUNTER register read taking fixed five PCLK16M clock cycles.

6.13.10 Registers

Base address	Peripheral	Instance	Secure mapping	DMA security	Description	Configuration
0x50014000 0x40014000	RTC	RTC0 : S RTC0 : NS	US	NA	Real time counter 0	
0x50015000 0x40015000	RTC	RTC1 : S RTC1 : NS	US	NA	Real time counter 1	

Table 62: Instances

Register	Offset	Security	Description
TASKS_START	0x000		Start RTC counter
TASKS_STOP	0x004		Stop RTC counter
TASKS_CLEAR	0x008		Clear RTC counter

[8] Assumes RTC runs continuously between these events.

Note: 32.768 kHz clock jitter is additional to the numbers provided above.

Register	Offset	Security	Description
TASKS_TRIGOVRFLW	0x00C		Set counter to 0xFFFFF0
SUBSCRIBE_START	0x080		Subscribe configuration for task START
SUBSCRIBE_STOP	0x084		Subscribe configuration for task STOP
SUBSCRIBE_CLEAR	0x088		Subscribe configuration for task CLEAR
SUBSCRIBE_TRIGOVRFLW	0x08C		Subscribe configuration for task TRIGOVRFLW
EVENTS_TICK	0x100		Event on counter increment
EVENTS_OVRFLW	0x104		Event on counter overflow
EVENTS_COMPARE[0]	0x140		Compare event on CC[0] match
EVENTS_COMPARE[1]	0x144		Compare event on CC[1] match
EVENTS_COMPARE[2]	0x148		Compare event on CC[2] match
EVENTS_COMPARE[3]	0x14C		Compare event on CC[3] match
PUBLISH_TICK	0x180		Publish configuration for event TICK
PUBLISH_OVRFLW	0x184		Publish configuration for event OVRFLW
PUBLISH_COMPARE[0]	0x1C0		Publish configuration for event COMPARE[0]
PUBLISH_COMPARE[1]	0x1C4		Publish configuration for event COMPARE[1]
PUBLISH_COMPARE[2]	0x1C8		Publish configuration for event COMPARE[2]
PUBLISH_COMPARE[3]	0x1CC		Publish configuration for event COMPARE[3]
SHORTS	0x200		Shortcuts between local events and tasks
INTENSET	0x304		Enable interrupt
INTENCLR	0x308		Disable interrupt
EVTEN	0x340		Enable or disable event routing
EVTENSET	0x344		Enable event routing
EVTENCLR	0x348		Disable event routing
COUNTER	0x504		Current counter value
PRESCALER	0x508		12-bit prescaler for counter frequency (32768/(PRESCALER+1)). Must be written when RTC is stopped.
CC[n]	0x540		Compare register n

Table 63: Register overview

6.13.10.1 TASKS_START

Address offset: 0x000

Start RTC counter

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_START			Start RTC counter
			Trigger	1	Trigger task

6.13.10.2 TASKS_STOP

Address offset: 0x004

Stop RTC counter

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_STOP			Stop RTC counter
			Trigger	1	Trigger task

6.13.10.3 TASKS_CLEAR

Address offset: 0x008

Clear RTC counter

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_CLEAR			Clear RTC counter
			Trigger	1	Trigger task

6.13.10.4 TASKS_TRIGOVRFLW

Address offset: 0x00C

Set counter to 0xFFFFF0

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	W	TASKS_TRIGOVRFLW			Set counter to 0xFFFFF0
			Trigger	1	Trigger task

6.13.10.5 SUBSCRIBE_START

Address offset: 0x080

Subscribe configuration for task START

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task START will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.13.10.6 SUBSCRIBE_STOP

Address offset: 0x084

Subscribe configuration for task STOP

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task STOP will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.13.10.7 SUBSCRIBE_CLEAR

Address offset: 0x088

Subscribe configuration for task CLEAR

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task CLEAR will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.13.10.8 SUBSCRIBE_TRIGOVRFLW

Address offset: 0x08C

Subscribe configuration for task TRIGOVRFLW

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that task TRIGOVRFLW will subscribe to
B	RW	EN			
			Disabled	0	Disable subscription
			Enabled	1	Enable subscription

6.13.10.9 EVENTS_TICK

Address offset: 0x100

Event on counter increment

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_TICK			Event on counter increment
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.13.10.10 EVENTS_OVRFLW

Address offset: 0x104

Event on counter overflow

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_OVRFLW			Event on counter overflow
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.13.10.11 EVENTS_COMPARE[n] (n=0..3)

Address offset: 0x140 + (n × 0x4)

Compare event on CC[n] match

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																																A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	EVENTS_COMPARE			Compare event on CC[n] match
			NotGenerated	0	Event not generated
			Generated	1	Event generated

6.13.10.12 PUBLISH_TICK

Address offset: 0x180

Publish configuration for event TICK

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event TICK will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.13.10.13 PUBLISH_OVRFLW

Address offset: 0x184

Publish configuration for event OVRFLW

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event OVRFLW will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.13.10.14 PUBLISH_COMPARE[n] (n=0..3)

Address offset: 0x1C0 + (n × 0x4)

Publish configuration for event COMPARE[n]

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID	B																												A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	CHIDX		[15..0]	Channel that event COMPARE[n] will publish to.
B	RW	EN			
			Disabled	0	Disable publishing
			Enabled	1	Enable publishing

6.13.10.15 SHORTS

Address offset: 0x200

Shortcuts between local events and tasks

6.13.10.16 INTENSET

Address offset: 0x304

Enable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													F	E	D	C															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	TICK			Write '1' to enable interrupt for event TICK
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	OVRFLW			Write '1' to enable interrupt for event OVRFLW
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C-F	RW	COMPARE[i] (i=0..3)			Write '1' to enable interrupt for event COMPARE[i]
			Set	1	Enable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.13.10.17 INTENCLR

Address offset: 0x308

Disable interrupt

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													F	E	D	C															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	TICK			Write '1' to disable interrupt for event TICK
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
B	RW	OVRFLW			Write '1' to disable interrupt for event OVRFLW
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
C-F	RW	COMPARE[i] (i=0..3)			Write '1' to disable interrupt for event COMPARE[i]
			Clear	1	Disable
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled

6.13.10.18 EVTEN

Address offset: 0x340

Enable or disable event routing

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													F	E	D	C															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	TICK			Enable or disable event routing for event TICK
			Disabled	0	Disable
			Enabled	1	Disable
B	RW	OVRFLW			Enable or disable event routing for event OVRFLW
			Disabled	0	Disable
			Enabled	1	Disable
C-F	RW	COMPARE[i] (i=0..3)			Enable or disable event routing for event COMPARE[i]
			Disabled	0	Disable
			Enabled	1	Disable

6.13.10.19 EVTENSET

Address offset: 0x344

Enable event routing

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													F	E	D	C															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	TICK			Write '1' to enable event routing for event TICK
			Disabled	0	Read: Disabled

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													F	E	D	C															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
			Enabled	1	Read: Enabled
			Set	1	Enable
B	RW	OVRFLW			Write '1' to enable event routing for event OVRFLW
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
			Set	1	Enable
C-F	RW	COMPARE[i] (i=0..3)			Write '1' to enable event routing for event COMPARE[i]
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
			Set	1	Enable

6.13.10.20 EVTENCLR

Address offset: 0x348

Disable event routing

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID													F	E	D	C															B	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	TICK			Write '1' to disable event routing for event TICK
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
			Clear	1	Disable
B	RW	OVRFLW			Write '1' to disable event routing for event OVRFLW
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
			Clear	1	Disable
C-F	RW	COMPARE[i] (i=0..3)			Write '1' to disable event routing for event COMPARE[i]
			Disabled	0	Read: Disabled
			Enabled	1	Read: Enabled
			Clear	1	Disable

6.13.10.21 COUNTER

Address offset: 0x504

Current counter value

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID									A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	R	COUNTER			Counter value

6.13.10.22 PRESCALER

Address offset: 0x508

12-bit prescaler for counter frequency (32768/(PRESCALER+1)). Must be written when RTC is stopped.

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID																					A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	PRESCALER			Prescaler value

6.13.10.23 CC[n] (n=0..3)

Address offset: 0x540 + (n × 0x4)

Compare register n

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID									A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
Reset 0x00000000	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

ID	Acce	Field	Value ID	Value	Description
A	RW	COMPARE			Compare value

6.13.11 Electrical specification

6.14 SAADC — Successive approximation analog-to-digital converter

The ADC is a differential successive approximation register (SAR) analog-to-digital converter.

Listed here are the main features of SAADC:

- 8/10/12-bit resolution, 14-bit resolution with oversampling
- Up to eight input channels
  - One channel per single-ended input and two channels per differential input
  - Scan mode can be configured with both single-ended channels and differential channels.
- Full scale input range (0 to VDD)
- Sampling triggered via a task from software or a PPI channel for full flexibility on sample frequency source from low power 32.768kHz RTC or more accurate 1/16MHz Timers
- One-shot conversion mode to sample a single channel
- Scan mode to sample a series of channels in sequence. Sample delay between channels is t_{ack} + t_{conv} which may vary between channels according to user configuration of t_{ack}.
- Support for direct sample transfer to RAM using EasyDMA
- Interrupts on single sample and full buffer events
- Samples stored as 16-bit 2’s complement values for differential and single-ended sampling
- Continuous sampling without the need of an external timer
- Internal resistor string
- Limit checking on the fly

6.14.1 Shared resources

The ADC can coexist with COMP and other peripherals using one of `AIN0`–`AIN7`, provided these are assigned to different pins.

It is not recommended to select the same analog input pin for both modules.

6.14.2 Overview

The ADC supports up to eight external analog input channels, depending on package variant. It can be operated in a one-shot mode with sampling under software control, or a continuous conversion mode with a programmable sampling rate.

The analog inputs can be configured as eight single-ended inputs, four differential inputs or a combination of these. Each channel can be configured to select `AIN0` to `AIN7` pins, or the `VDD` pin. Channels can be sampled individually in one-shot or continuous sampling modes, or, using scan mode, multiple channels can be sampled in sequence. Channels can also be oversampled to improve noise performance.

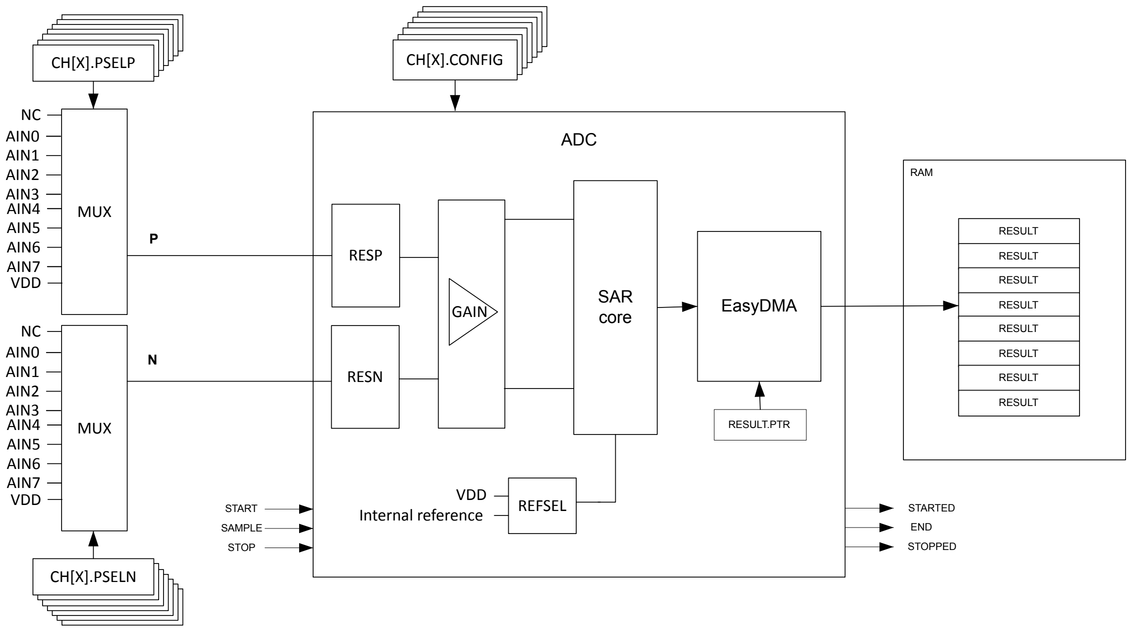


Figure 62: Simplified ADC block diagram

Internally, the ADC is always a differential analog-to-digital converter, but by default it is configured with single-ended input in the MODE field of the CH[n].CONFIG register. In single-ended mode, the negative input will be shorted to ground internally.

The assumption in single-ended mode is that the internal ground of the ADC is the same as the external ground that the measured voltage is referred to. The ADC is thus sensitive to ground bounce on the PCB in single-ended mode. If this is a concern we recommend using differential measurement.

6.14.3 Digital output

The output result of the ADC depends on the settings in the CH[n].CONFIG and RESOLUTION registers as follows:

$$\text{RESULT} = [V(P) - V(N)] * \text{GAIN/REFERENCE} * 2^{(\text{RESOLUTION} - m)}$$

where

V(P)

is the voltage at input P

V(N)

is the voltage at input N

GAIN

is the selected gain setting

REFERENCE	
	is the selected reference voltage

and m=0 if CONFIG.MODE=SE, or m=1 if CONFIG.MODE=Diff.

The result generated by the ADC will deviate from the expected due DC errors like offset, gain, differential non-linearity (DNL), and integral non-linearity (INL). See Electrical specification for details on these parameters. The result can also vary due to AC errors like non-linearities in the GAIN block, settling errors due to high source impedance and sampling jitter. For battery measurement the DC errors are most noticeable.

The ADC has a wide selection of gains controlled in the GAIN field of the CH[n].CONFIG register. If CH[n].CONFIG.REFSEL=0, the input range of the ADC core is nominally ±0.6 V differential and the input must be scaled accordingly.

The ADC has a temperature dependent offset. If the ADC is to operate over a large temperature range, we recommend running CALIBRATEOFFSET at regular intervals. The CALIBRATEDONE event will be fired when the calibration has been completed. Note that the DONE and RESULTDONE events will also be generated.

6.14.4 Analog inputs and channels

Up to eight analog input channels, CH[n](n=0..7), can be configured.

See Shared resources on page 195 for shared input with comparators.

Any one of the available channels can be enabled for the ADC to operate in one-shot mode. If more than one CH[n] is configured, the ADC enters scan mode.

An analog input is selected as a positive converter input if CH[n].PSELP is set, setting CH[n].PSELP also enables the particular channel.

An analog input is selected as a negative converter input if CH[n].PSELN is set. The CH[n].PSELN register will have no effect unless differential mode is enabled, see MODE field in CH[n].CONFIG register.

If more than one of the CH[n].PSELP registers is set, the device enters scan mode. Input selections in scan mode are controlled by the CH[n].PSELP and CH[n].PSELN registers, where CH[n].PSELN is only used if the particular scan channel is specified as differential, see MODE field in CH[n].CONFIG register.

Important: Channels selected for COMP cannot be used at the same time for ADC sampling, though channels not selected for use by these blocks can be used by the ADC.

6.14.5 Operation modes

The ADC input configuration supports one-shot mode, continuous mode and scan mode.

Scan mode and oversampling cannot be combined.

6.14.5.1 One-shot mode

One-shot operation is configured by enabling only one of the available channels defined by CH[n].PSELP, CH[n].PSELN, and CH[n].CONFIG registers.

Upon a SAMPLE task, the ADC starts to sample the input voltage. The CH[n].CONFIG.TACQ controls the acquisition time.

A DONE event signals that one sample has been taken.

In this mode, the RESULTDONE event has the same meaning as DONE when no oversampling takes place. Note that both events may occur before the actual value has been transferred into RAM by EasyDMA. For more information, see EasyDMA on page 199.

6.14.5.2 Continuous mode

Continuous sampling can be achieved by using the internal timer in the ADC, or triggering the SAMPLE task from one of the general purpose timers through the PPI.

Care shall be taken to ensure that the sample rate fulfils the following criteria, depending on how many channels are active:

$f_{SAMPLE} < 1/[t_{ACQ} + t_{conv}]$

The SAMPLERATE register can be used as a local timer instead of triggering individual SAMPLE tasks. When SAMPLERATE.MODE is set to Timers, it is sufficient to trigger SAMPLE task only once in order to start the SAADC and triggering the STOP task will stop sampling. The SAMPLERATE.CC field controls the sample rate.

The SAMPLERATE timer mode cannot be combined with SCAN mode, and only one channel can be enabled in this mode.

A DONE event signals that one sample has been taken.

In this mode, the RESULTDONE event has the same meaning as DONE when no oversampling takes place. Note that both events may occur before the actual value has been transferred into RAM by EasyDMA.

6.14.5.3 Oversampling

An accumulator in the ADC can be used to average noise on the analog input. In general, oversampling improves the signal-to-noise ratio (SNR). Oversampling, however, does not improve the integral non-linearity (INL), or differential non-linearity (DNL).

Oversampling and scan should not be combined, since oversampling and scan will average over input channels.

The accumulator is controlled in the OVERSAMPLE register. The SAMPLE task must be set $2^{OVERSAMPLE}$ number of times before the result is written to RAM. This can be achieved by:

- Configuring a fixed sampling rate using the local timer or a general purpose timer and PPI to trigger a SAMPLE task
- Triggering SAMPLE $2^{OVERSAMPLE}$ times from software
- Enabling BURST mode

CH[n].CONFIG.BURST can be enabled to avoid setting SAMPLE task $2^{OVERSAMPLE}$ times. With BURST = 1 the ADC will sample the input $2^{OVERSAMPLE}$ times as fast as it can (actual timing: $<(t_{ACQ}+t_{CONV})\times2^{OVERSAMPLE}$). Thus, for the user it will just appear like the conversion took a bit longer time, but other than that, it is similar to one-shot mode.

A DONE event signals that one sample has been taken.

In this mode, the RESULTDONE event signals that enough conversions have taken place for an oversampled result to get transferred into RAM. Note that both events may occur before the actual value has been transferred into RAM by EasyDMA.

6.14.5.4 Scan mode

A channel is considered enabled if CH[n].PSELP is set. If more than one channel, CH[n], is enabled, the ADC enters scan mode.

In scan mode, one SAMPLE task will trigger one conversion per enabled channel. The time it takes to sample all channels is:

```
Total time < Sum(CH[x].tACQ+tCONV), x=0..enabled channels
```

A DONE event signals that one sample has been taken.