# nRF51 Series Reference Manual

Version 3.0.1

The nRF51 series offers a range of ultra-low power System on Chip solutions for your 2.4 GHz wireless products. With the nRF51 series you have a diverse selection of devices including those with embedded *Bluetooth*® low energy and/or ANT™ protocol stacks as well as open devices enabling you to develop your own proprietary wireless stack and ecosystem.

The nRF51 series combines Nordic Semiconductor's leading 2.4 GHz transceiver technology with a powerful but low power ARM® Cortex™-M0 core, a range of peripherals and memory options. The pin and code compatible devices of the nRF51 series offer you the most flexible platform for all your 2.4 GHz wireless applications.


2016-12-06

# Contents

# 1 Revision history

| Date | Version | Description |
|---|---|---|
| December 2016 | 3.0.1 | **Updated content:**<br>• Section 5.2 Instantiation<br>• Section 12.1.3 DC/DC converter setup<br>• Section 12.1.15 Soft reset<br>• Section 12.1.19 Reset behavior<br>• Section 13.1.2 LFCLK Clock controller<br>• Section 13.3, Table 71: LFCLKSRC<br>• Section 13.3, Table 73: XTALFREQ<br>• Section 19.2, Table 152: Instances<br>• Section 29.11, Table 287: Baudrate<br>• Chapter 31, Analog to Digital Converter |
| September 2014 | 3.0b | **Added content:**<br>• Software Interrupts chapter<br>**Updated content:**<br>• Power chapter |
| November 2013 | 2.1 | **Updated content:**<br>• Table 6 on page 19.<br>• Figure 72 on page 181<br>• Section31.4.5 on page 185 |

# 2 About this document

This reference manual is a functional description of all the modules and peripherals supported by the nRF51 series and subsequently, is a common document for all nRF51 System on Chip (SoC) devices.

> **Note:** nRF51 SoC devices may not support all the modules and peripherals described in this document and some of their implemented modules may have a reduced feature set. Please refer to the individual nRF51 device product specification for details on the supported feature set, electrical and mechanical specifications, and application specific information.

## 2.1 Peripheral naming and abbreviations

Every peripheral has a unique name or an abbreviation constructed by a single word, e.g. TIMER. This name is indicated in parentheses in the peripheral chapter heading. This name will be used in CMSIS 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.

## 2.2 Register tables

Individual registers are described using register tables. These tables are built up of two sections. The first three rows, which are shaded blue, describe the position and size of the different fields in the register. The following rows, beginning with the row shaded green, describes the fields in more detail.

### 2.2.1 Fields and values

The **Id (Field Id)** row specifies which bits that belong to the different fields in the register.

A blank space means that the field is reserved and that it is read as undefined and 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.

If a field has enumerated values, then every value will be identified with a unique value Id in the **Value Id** column. Single-bit bit-fields may however omit the “Value Id” when values can be substituted with a Boolean type enumerator range, for example, True, False; Disable, Enable, and On, Off, and so on.

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

- Individual enumerated values, for example, 1, 3, 9.
- Range values, e.g. [0..4], that is, 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 “..”.

When a row in a register table contains the word **Deprecated** it means this is an attribute applied to a feature to indicate that it should not be used for new designs.

#### Table 1: Example register table

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | WEN | | | Program memory access mode. It is strongly recommended to only activate erase and write modes when they are actively used. |
| | | | Ren | 0 | Read only access |
| | | | Wen | 1 | Write 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 | A |
| Reset | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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** | | | | | | | | | | | | | | | | | | | | | | | |
| | | | Een | 2 | | | | | | | | Erase enabled | | | | | | | | | | | | | | | | | | | | | | | |

# 3 System overview

## 3.1 Summary

The nRF51 series of System on Chip (SoC) devices embed a powerful yet low power ARM® Cortex™-M0 processor with our industry leading 2.4 GHz RF transceivers. In combination with the very flexible orthogonal power management system and a Programmable Peripheral Interconnect (PPI) event system, the nRF51 series enables you to make ultra-low power wireless solutions.

The nRF51 series offers pin compatible device options for *Bluetooth* low energy, proprietary 2.4 GHz, and ANT™ solutions giving you the freedom to develop your wireless system using the technology that suits your application the best. Our unique memory and hardware resource protection system allows you to develop applications on devices with embedded protocol stacks running on the same processor without any need to link in the stack or strenuous testing to avoid application and stack from interfering with each other.

## 3.2 Block diagram

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

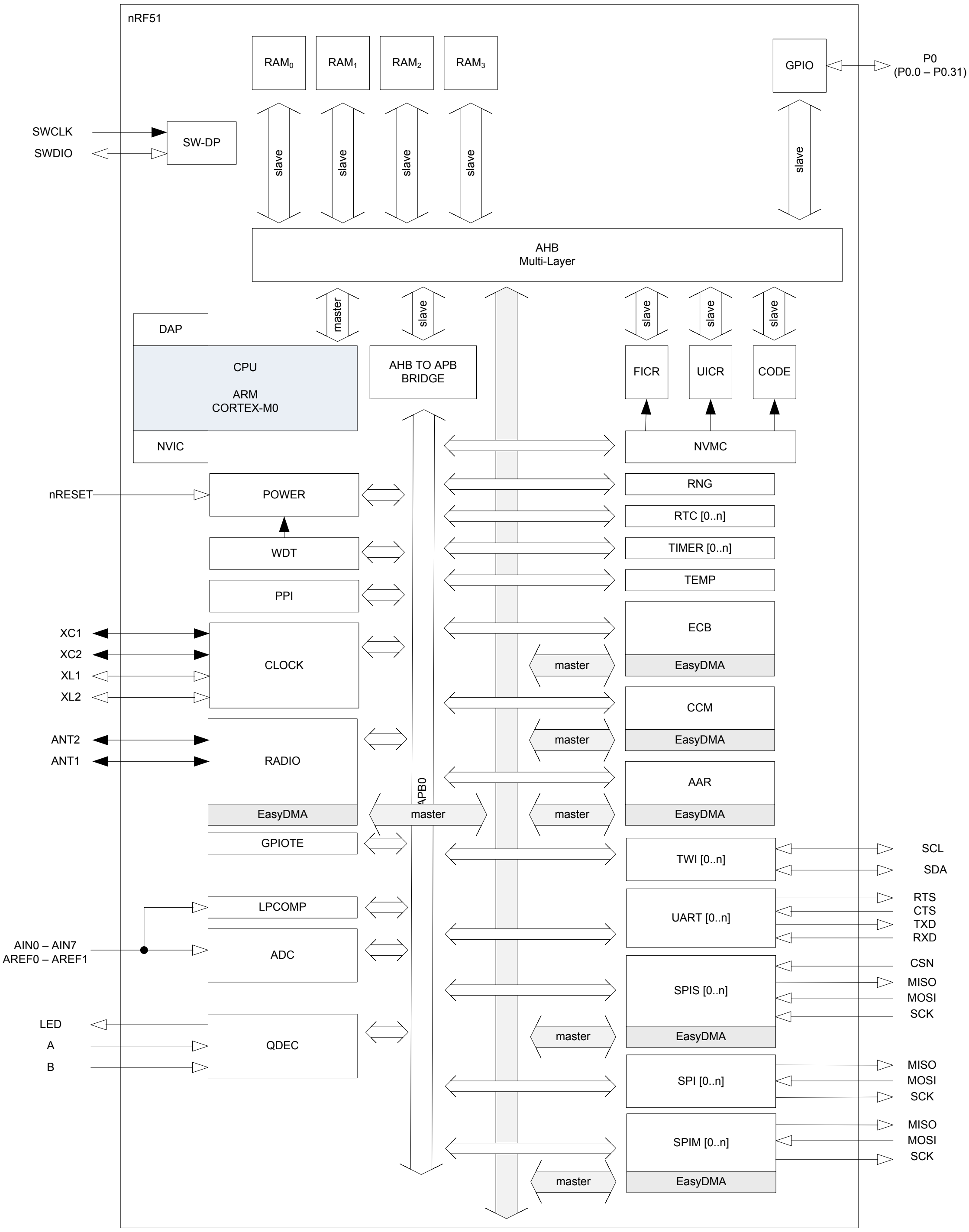



**Figure 1: Block diagram**

## 3.3 System blocks

This section contains descriptions of the main blocks that make up the nRF51 series.

### 3.3.1 ARM® Cortex™-M0

A low power ARM® Cortex™-M0 32 bit CPU is embedded in all nRF51 series devices. The ARM® Cortex™-M0 has a 16 bit instruction set with 32 bit extensions (**Thumb-2® technology**) that delivers high density code with a small memory footprint. By using a single-cycle 32 bit multiplier, a 3-stage pipeline, and a Nested Vector Interrupt Controller (NVIC), the ARM® Cortex™-M0 CPU makes program execution simple and highly efficient.

The ARM® Cortex Microcontroller Software Interface Standard (CMSIS) hardware abstraction layer for the ARM® Cortex-M processor series is implemented and available for M0 CPU. Code is forward compatible with ARM® Cortex-M3 and ARM® Cortex-M4 based devices.

### 3.3.2 2.4 GHz radio

The nRF51 series ultra-low power 2.4 GHz GFSK RF transceiver is designed and optimized to operate in the worldwide ISM frequency band at 2.400 GHz to 2.4835 GHz. Configurable radio modulation modes and packet structure makes the transceiver interoperable with *Bluetooth* low energy (BLE), ANT™, Gazell, Enhanced Shockburst™, and a range of other 2.4 GHz protocol implementations.

The transceiver receives and transmits data directly to and from system memory. It is stored in clear text even when encryption is enabled, so packet data management is flexible and efficient.

### 3.3.3 Power management

The nRF51 series power management system is orthogonal and highly flexible with only simple ON or OFF modes governing a whole device. In System OFF mode, everything is powered down but sections of the RAM can be retained. The device state can be changed to System ON through reset or wake up from all GPIOs. When in System ON mode, all functional blocks are accessible with each functional block remaining in IDLE mode and only entering RUN mode when required.

### 3.3.4 PPI system

The Programmable Peripheral Interconnect (PPI) enables different peripherals to interact autonomously with each other using tasks and events without use of the CPU. The PPI provides a mechanism to automatically trigger a task in one peripheral as a result of an event occurring in another. A task is connected to an event through a PPI channel.

### 3.3.5 Debugger support

The 2 pin Serial Wire Debug interface (provided as a part of the Debug Access Port, DAP) offers a flexible and powerful mechanism for non-intrusive program code debugging. This includes adding breakpoints in the code and performing single stepping.

# 4 CPU

A low power ARM® Cortex™-M0 32 bit CPU is embedded in all nRF51 series devices. The ARM® Cortex™-M0 has a 16 bit instruction set with 32 bit extensions (**Thumb-2® technology**) that delivers high density code with a small memory footprint. By using a single-cycle 32 bit multiplier, a 3-stage pipeline, and a Nested Vector Interrupt Controller (NVIC), the ARM® Cortex™-M0 CPU makes program execution simple and highly efficient.

The data alignment in nRF51 implementation is Little Endian.

The ARM® Cortex Microcontroller Software Interface Standard (CMSIS) hardware abstraction layer for the ARM® Cortex-M processor series is implemented and available for M0 CPU. Code is forward compatible with ARM® Cortex-M3 based devices.

For further information on the embedded ARM® Cortex™-M0 CPU, see *ARM Cortex M0*.

# 5 Memory

## 5.1 Functional description

All memory blocks and registers are placed in a common memory map.

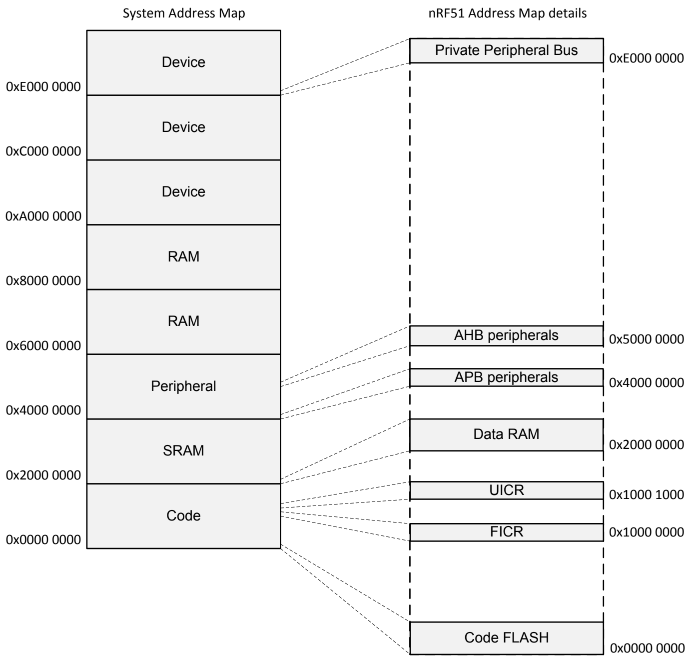



**Figure 2: Memory map**

### 5.1.1 Memory categories

There are three main categories of memory:

- Code memory
- Random Access Memory (RAM)
- Peripheral registers (PER)

In addition, there is one information block (FICR) containing read only parameters describing configuration details of the device and another information block (UICR) that can be configured by the user.

### 5.1.2 Memory types

The various memory categories can have one of the following memory types:

- Volatile memory (VM)
- Non-volatile memory (NVM)

Volatile memory is a type of memory that will lose its contents when the chip loses power. This memory type can be read/written an unlimited number of times by the CPU.

Non-volatile memory is a type of memory that can retain stored information even when the chip loses power. This memory type can be read an unlimited number of times by the CPU, but have restrictions on the number of times it can be written and erased[1] and also on how it can be written. Writing to non-volatile memory is managed by the Non Volatile Memory Controller (NVMC).

[1] See product specification for more information

### 5.1.3 Code memory

The code memory is normally used for storing the program executed by the CPU, but can also be used for storing data constants that are retained when the chip loses power.

The code memory is non-volatile.

### 5.1.4 Random Access Memory

All RAM is volatile and always loses its content when the chip loses power.

Whether the RAM content is lost in System OFF power saving mode is dependent on the settings in the RAMON register in the POWER peripheral.

The system includes the following RAM (Random Access Memory) regions:

- Data RAM

The Data RAM region is located in the SRAM segment of the System Address Map. It is possible to execute code from this region.

The RAM interface is divided into multiple RAM AHB (AMBA High-performance Bus) slaves.

Each RAM AHB slave is connected to one 4 kbyte RAM section, see Section 0 in *Figure 3: RAM mapping* on page 16.

A RAM block is defined as two RAM sections as illustrated in *Figure 3: RAM mapping* on page 16.

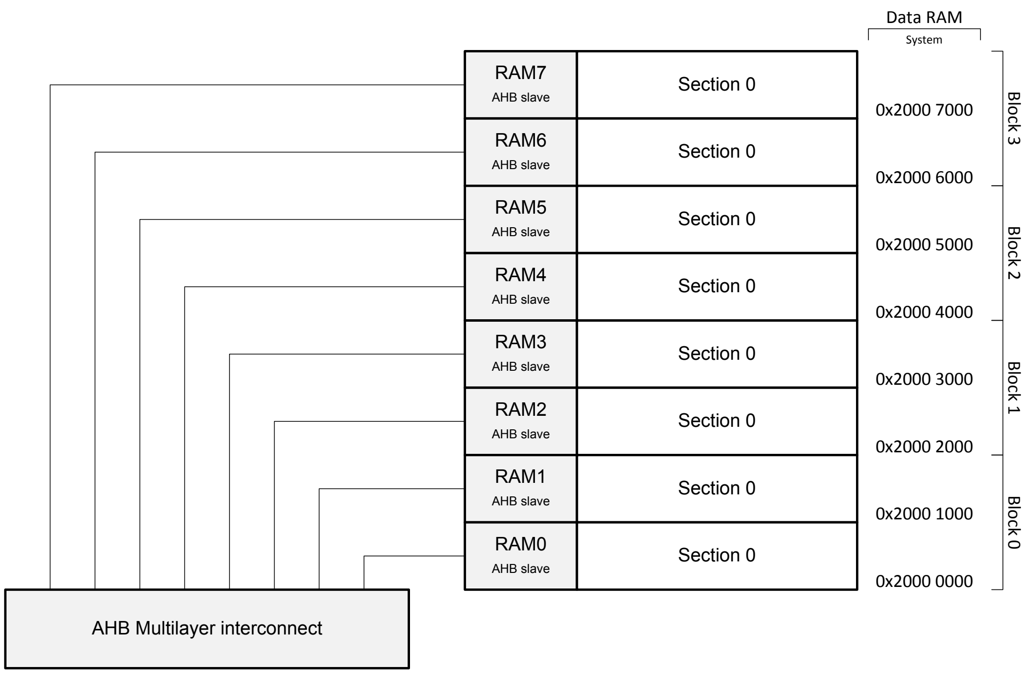



**Figure 3: RAM mapping**

See product specification for more information about how many blocks and RAM AHB slaves are implemented.

### 5.1.5 Peripheral registers

The peripheral registers are registers used for interfacing to peripheral units such as timers, the radio, the ADC, and so on.

Most peripherals feature an ENABLE register. Unless otherwise specified in the relevant chapter, the peripheral registers (in particular the PSEL registers) shall be configured prior to enabling the peripheral.

When switching from one peripheral to another sharing the same base address (see **Instantiation** below to find for which peripherals this is the case), one shall disable the other peripheral currently using the base address, configure the new settings, and then enable the new peripheral.

Note that tasks and events cannot be used prior to enabling the peripheral.

Some peripherals feature a POWER register. This register is not required to be used unless specifically required by a PAN (Product Anomaly Notice).

## 5.2 Instantiation

### Table 2: Instantiation table

| ID | Base address | Peripheral | Instance | Description |
|---|---|---|---|---|
| 0 | 0x40000000 | POWER | POWER | Power Control |
| 0 | 0x40000000 | CLOCK | CLOCK | Clock control |
| 0 | 0x40000000 | MPU | MPU | Memory Protection Unit |
| 1 | 0x40001000 | RADIO | RADIO | 2.4 GHz radio |
| 2 | 0x40002000 | UART | UART0 | Universal Asynchronous Receiver/Transmitter |
| 3 | 0x40003000 | SPI | SPI0 | SPI master 0 |
| 3 | 0x40003000 | TWI | TWI0 | Two-wire interface master 0 |
| 4 | 0x40004000 | TWI | TWI1 | Two-wire interface master 1 |
| 4 | 0x40004000 | SPI | SPI1 | SPI master 1 |
| 4 | 0x40004000 | SPIS | SPIS1 | SPI slave 1 |
| 6 | 0x40006000 | GPIOTE | GPIOTE | GPIO tasks and events |
| 7 | 0x40007000 | ADC | ADC | Analog to digital converter |
| 8 | 0x40008000 | TIMER | TIMER0 | Timer 0<br><br>This timer instance has 4 CC registers<br><br>This timer instance supports BITMODE = 08Bit, 16Bit, 24Bit and 32Bit |
| 9 | 0x40009000 | TIMER | TIMER1 | Timer 1<br><br>This timer instance has 4 CC registers<br><br>This timer instance supports BITMODE = 08Bit and 16Bit |
| 10 | 0x4000A000 | TIMER | TIMER2 | Timer 2<br><br>This timer instance has 4 CC registers<br><br>This timer instance supports BITMODE = 08Bit and 16Bit |
| 11 | 0x4000B000 | RTC | RTC0 | Real time counter 0 |
| 12 | 0x4000C000 | TEMP | TEMP | Temperature Sensor |
| 13 | 0x4000D000 | RNG | RNG | Random Number Generator |
| 14 | 0x4000E000 | ECB | ECB | AES ECB Mode Encryption |
| 15 | 0x4000F000 | AAR | AAR | Accelerated Address Resolver |
| 15 | 0x4000F000 | CCM | CCM | AES CCM Mode Encryption |
| 16 | 0x40010000 | WDT | WDT | Watchdog Timer |
| 17 | 0x40011000 | RTC | RTC1 | Real time counter 1 |
| 18 | 0x40012000 | QDEC | QDEC | Quadrature decoder |
| 19 | 0x40013000 | LPCOMP | LPCOMP | Low power comparator |
| 20 | 0x40014000 | SWI | SWI0 | Software interrupt 0 |
| 21 | 0x40015000 | SWI | SWI1 | Software interrupt 1 |
| 22 | 0x40016000 | SWI | SWI2 | Software interrupt 2 |
| 23 | 0x40017000 | SWI | SWI3 | Software interrupt 3 |
| 24 | 0x40018000 | SWI | SWI4 | Software interrupt 4 |
| 25 | 0x40019000 | SWI | SWI5 | Software interrupt 5 |
| 30 | 0x4001E000 | NVMC | NVMC | Non Volatile Memory Controller |
| 31 | 0x4001F000 | PPI | PPI | PPI controller |
| N/A | 0x10000000 | FICR | FICR | Factory Information Configuration |
| N/A | 0x10001000 | UICR | UICR | User Information Configuration |
| N/A | 0x50000000 | GPIO | GPIO | General purpose input and output |

# 6 Non-Volatile Memory Controller (NVMC)

## 6.1 Functional description

The Non-volatile Memory Controller (NVMC) is used for writing and erasing Non-volatile Memory (NVM).

Before a write can be performed the NVM must be enabled for writing in CONFIG.WEN. Similarly, before an erase can be performed the NVM must be enabled for erasing in CONFIG.EEN. The user must make sure that writing and erasing is not enabled at the same time, failing to do so may result in unpredictable behavior.

### 6.1.1 Writing to the NVM

When writing is enabled, the NVM is written by writing a word to a word aligned address in the CODE or UICR. The NVMC is only able to write bits in the NVM that are erased, that is, set to '1'.

The same address in the NVM can only be written $n_{WRITE}$ number of times before an erase must be performed using PAGEERASE or ERASEALL.

The time it takes to write a word to the NVM is specified by $t_{WRITE}$ in the product specification. The CPU is halted while the NVMC is writing to the NVM.

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

### 6.1.2 Writing to User Information Configuration Registers

UICR registers are written as ordinary non-volatile memory. After the UICR has been written, the new UICR configuration will only take effect after a reset.

### 6.1.3 Erase all

When erase is enabled, the whole CODE and UICR can be erased in one operation by using the ERASEALL register. ERASEALL will not erase the Factory Information Configuration Registers (FICR).

The time it takes to perform an ERASEALL command is specified by $t_{ERASEALL}$ in the product specification. The CPU is halted while the NVMC performs the erase operation.

### 6.1.4 Erasing a page in code region 1

When erase is enabled, the NVM can be erased page by page using the ERASEPAGE register or the ERASEPCR1 register. After erasing a NVM page all bits in the page are set to '1'. The time it takes to erase a page is specified by $t_{PAGEERASE}$ in the product specification. The CPU is halted while the NVMC performs the erase operation. See *UICR* chapter for more information.

### 6.1.5 Erasing a page in code region 0

ERASEPCR0 is used to erase a page in code region 0. The ERASEPCR0 register can only be accessed from a program running in code region 0.

To enable non-volatile storage for program running in code region 0, it is possible for this program to erase and re-write any code page it designates for this purpose within code region 0. The ERASEPCR0 can be used for this purpose. The ERASEPCR0 register has a restriction on its use, enforced by the MPU, where only code running from code region 0 can write to it. It is possible for a program running from code region 0 to erase a page in code region 1 using ERASEPCR1.

The time it takes to erase a page is specified by $t_{PAGEERASE}$ in the product specification.

# 6.2 Register Overview

### Table 3: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x4001E000 | NVMC | NVMC | Non Volatile Memory Controller |

### Table 4: Register Overview

| Register | Offset | Description |
|---|---|---|
| Registers | | |
| *READY* | 0x400 | Ready flag |
| *CONFIG* | 0x504 | Configuration register |
| *ERASEPAGE* | 0x508 | Register for erasing a page in Code area |
| *ERASEPCR1* | 0x508 | Register for erasing a page in Code region 1. Equivalent to ERASEPAGE. |
| *ERASEALL* | 0x50C | Register for erasing all non-volatile user memory |
| *ERASEPCR0* | 0x510 | Register for erasing a page in Code region 0 |
| *ERASEUICR* | 0x514 | Register for erasing User Information Configuration Registers |

# 6.3 Register Details

### Table 5: READY

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | READY | | | NVMC is ready or busy |
| | | | Busy | 0 | NVMC is busy (on-going write or erase operation) |
| | | | Ready | 1 | NVMC is ready |

### Table 6: CONFIG

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | WEN | | | Program memory access mode. It is strongly recommended to only activate erase and write modes when they are actively used. |
| | | | Ren | 0 | Read only access |
| | | | Wen | 1 | Write Enabled |
| | | | Een | 2 | Erase enabled |

### Table 7: ERASEPAGE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ERASEPAGE | | | Register for starting erase of a page in Code region 1 The value is the address to the page to be erased. (Addresses of first word in page). Note that code erase has to be enabled by CONFIG.EEN before the page can be erased. See product specification for information about the total code size of the device you are using. Attempts to erase pages that are outside the code area may result in undesirable behaviour, e.g. the wrong page may be erased. |

### Table 8: ERASEPCR1

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ERASEPCR1 | | | Register for erasing a page in Code region 1. Equivalent to ERASEPAGE. |

### Table 9: ERASEALL

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ERASEALL | | | Erase all non-volatile memory including UICR registers. Note that code erase has to be enabled by CONFIG.EEN before the UICR can be erased. |
| | | | NoOperation | 0 | No operation |
| | | | Erase | 1 | Start chip erase |

### Table 10: ERASEPCR0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ERASEPCR0 | | | Register for starting erase of a page in Code region 0 The value is the address to the page to be erased (address of first word in page). Only page addresses in Code region 0 are allowed. This register can only be accessed from a program running in Code memory region 0. A hard fault will be generated if the register is attempted accessed from a program in RAM or Code memory region 1. Writing to ERASEPCR0 from the Serial Wire Debug (SWD) will have no effect. CONFIG.EEN has to be set to enable erase. See product specification for information about the total code size of the device you are using. Attempts to erase pages that are outside the code area may result in undesirable behaviour, e.g. the wrong page may be erased. |

### Table 11: ERASEUICR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ERASEUICR | | | Register starting erase of all User Information Configuration Registers. Note that code erase has to be enabled by CONFIG.EEN before the UICR can be erased. |
| | | | NoOperation | 0 | No operation |
| | | | Erase | 1 | Start erase of UICR |

# 7 Factory Information Configuration Registers (FICR)

## 7.1 Functional description

Factory Information Configuration Registers are pre-programmed in factory and cannot be erased by the user. These registers contain chip specific information and configuration.

## 7.2 Override parameters

Factory Information Configuration Registers contain override parameters set during device calibration in production, which need to replace default settings in the RADIO. Override parameters and the RADIO mode they are used for varies between nRF51 devices. Read the OVERRIDDEN register to determine if the FICR contains override parameters for the radio mode you are going to use. If the FICR contains override parameters, they must be copied to the radio OVERRIDE registers before enabling the radio in that mode.

## 7.3 Register Overview

### Table 12: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x10000000 | FICR | FICR | Factory Information Configuration Registers |

### Table 13: Register Overview

| Register | Offset | Description | |
|---|---|---|---|
| Registers | | | |
| *CODEPAGESIZE* | 0x010 | Code memory page size | |
| *CODESIZE* | 0x014 | Code memory size | |
| *CLENR0* | 0x028 | Length of Code region 0 in bytes | *Deprecated* |
| *PPFC* | 0x02C | Pre-programmed factory Code present | *Deprecated* |
| *NUMRAMBLOCK* | 0x034 | Number of individually controllable RAM blocks | |
| *SIZERAMBLOCKS* | 0x038 | RAM block size, in bytes | |
| *SIZERAMBLOCK[0]* | 0x038 | Size of RAM block 0, in bytes | *Deprecated* |
| *SIZERAMBLOCK[1]* | 0x03C | Size of RAM block 1, in bytes | *Deprecated* |
| *SIZERAMBLOCK[2]* | 0x040 | Size of RAM block 2, in bytes | *Deprecated* |
| *SIZERAMBLOCK[3]* | 0x044 | Size of RAM block 3, in bytes | *Deprecated* |
| *CONFIGID* | 0x05C | Configuration identifier | |
| *DEVICEID[0]* | 0x060 | Device identifier | |
| *DEVICEID[1]* | 0x064 | Device identifier | |
| *ER[0]* | 0x080 | Encryption Root, word 0 | |
| *ER[1]* | 0x084 | Encryption Root, word 1 | |
| *ER[2]* | 0x088 | Encryption Root, word 2 | |
| *ER[3]* | 0x08C | Encryption Root, word 3 | |
| *IR[0]* | 0x090 | Identity Root, word 0 | |
| *IR[1]* | 0x094 | Identity Root, word 1 | |
| *IR[2]* | 0x098 | Identity Root, word 2 | |
| *IR[3]* | 0x09C | Identity Root, word 3 | |
| *DEVICEADDRTYPE* | 0x0A0 | Device address type | |
| *DEVICEADDR[0]* | 0x0A4 | Device address 0 | |
| *DEVICEADDR[1]* | 0x0A8 | Device address 1 | |
| *OVERRIDEEN* | 0x0AC | Override enable | |
| *NRF_1MBIT[0]* | 0x0B0 | Override value for NRF_1MBIT mode | |
| *NRF_1MBIT[1]* | 0x0B4 | Override value for NRF_1MBIT mode | |
| *NRF_1MBIT[2]* | 0x0B8 | Override value for NRF_1MBIT mode | |
| *NRF_1MBIT[3]* | 0x0BC | Override value for NRF_1MBIT mode | |
| *NRF_1MBIT[4]* | 0x0C0 | Override value for NRF_1MBIT mode | |
| *BLE_1MBIT[0]* | 0x0EC | Override value for BLE_1MBIT mode | |
| *BLE_1MBIT[1]* | 0x0F0 | Override value for BLE_1MBIT mode | |
| *BLE_1MBIT[2]* | 0x0F4 | Override value for BLE_1MBIT mode | |
| *BLE_1MBIT[3]* | 0x0F8 | Override value for BLE_1MBIT mode | |
| *BLE_1MBIT[4]* | 0x0FC | Override value for BLE_1MBIT mode | |

# 7.4 Register Details

## Table 14: CODEPAGESIZE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | CODEPAGESIZE | | | Code memory page size |

## Table 15: CODESIZE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | CODESIZE | | | Code memory size in number of pages<br>Total code space is: CODEPAGESIZE * CODESIZE |

## Table 16: CLENR0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | CLENR0 | | [0..N] | Length of code region 0 in bytes<br>Length of code region 0 in bytes. The value must be a multiple of "Code page size" bytes CODEPAGESIZE. This register is only used when pre-programmed factory Code is present on the chip, see PPFC. N (max value) is (CODEPAGESIZE * ( CODESIZE - 1 )). This register can only be written if content is 0xFFFFFFFF. Value after mass erase of flash is 0xFFFFFFFF, this value is interpreted as 0. |

## Table 17: PPFC

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | PPFC | | | Pre-programmed factory Code present or not |
| | | | NotPresent | 0xFF | Not present |
| | | | Present | 0x00 | Present |

## Table 18: NUMRAMBLOCK

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | NUMRAMBLOCK | | | Number of individually controllable RAM blocks |

## Table 19: SIZERAMBLOCKS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | SIZERAMBLOCKS | | | RAM block size, in bytes |

## Table 20: SIZERAMBLOCK[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | SIZERAMBLOCK | | | Size of RAM block n, in bytes |

## Table 21: CONFIGID

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | HWID | | | Identification number for the HW |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description | |
|---|---|---|---|---|---|---|
| B | R | FWID | | | Identification number for the FW that is pre-loaded into the chip | *Deprecated* |

## Table 22: DEVICEID[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | DEVICEID | | | 64 bit unique device identifier<br>DEVICEID[0] contains the least significant bits of the device identifier. DEVICEID[1] contains the most significant bits of the device identifier. |

## Table 23: ER[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | ER | | | Encryption Root, word n |

## Table 24: IR[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | IR | | | Identity Root, word n |

## Table 25: DEVICEADDRTYPE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | DEVICEADDRTYPE | | | Device address type |
| | | | Public | 0 | Public address |
| | | | Random | 1 | Random address |

## Table 26: DEVICEADDR[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | DEVICEADDR | | | 48 bit device address<br>DEVICEADDR[0] contains the least significant bits of the device address. DEVICEADDR[1] contains the most significant bits of the device address. Only bits [15:0] of DEVICEADDR[1] are used. |

## Table 27: OVERRIDEEN

Indicates whether or not a particular RADIO MODE setting must be overridden via the OVERRIDEn registers in the RADIO.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | NRF_1MBIT | | | Override default values for NRF_1MBIT mode |
| | | | Override | 0 | Override |
| | | | NotOverride | 1 | Do not override |
| D | R | BLE_1MBIT | | | Override default values for BLE_1MBIT mode |
| | | | Override | 0 | Override |
| | | | NotOverride | 1 | Do not override |

## Table 28: NRF_1MBIT[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | OVERRIDE | | | Override values for 1Mbit proprietary 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 | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
| Reset | | | | | 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** | **RW** | **Field** | **Value Id** | **Value** | **Description** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | Value to be written to RADIO.OVERRIDE[n] register if OVERRIDEEN is set. If override values are enabled for more than one mode the RADIO.OVERRIDE[n] registers has to be updated every time RADIO.MODE is changed. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

### Table 29: BLE_1MBIT[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 | A | A | A | A | A | A | A | A |
| Reset | | | | | 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** | **RW** | **Field** | **Value Id** | **Value** | **Description** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| A | R | OVERRIDE | | | Override value for 1 Mbit BLE mode<br>Value to be written to RADIO.OVERRIDE[n] register if OVERRIDEEN is set. If override values are enabled for more than one mode the RADIO.OVERRIDE[n] registers has to be updated every time RADIO.MODE is changed. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

# 8 User Information Configuration Registers (UICR)

## 8.1 Functional description

The User Information Configuration Registers (UICRs) are NVM registers for configuring user specific settings.

Code readback protection of the whole code area, or a part of the code area can be configured and enabled in the UICR. The UICR can only be erased by using ERASEALL.

The code area can be divided into two regions, code region 0 (CR0) and code region 1 (CR1). Code region 0 starts at address 0x00000000 and stretches into the code area as specified in the CLENR0 register. The area above CLENR0 will then be defined as code region 1. If CLENR0 is not configured, that is, has the value 0xFFFFFFFF, the whole code area will be defined as code region 1 (CR1).

Code running from code region 1 will not be able to write to code region 0. Additionally, the content of code region 0 cannot be read from code running in code region 1 or through the SWD interface if code region 0 is readback protected, see PR0 in RBPCONF.

The main readback protection mechanism that will protect the whole code, that is, both code region 0 and code region 1, is also configured through the UICR.

The PAGEERASE command in NVMC will only work for code region 1. See *NVMC* chapter for information on how to erase and program the code area and the *UICR*.

## 8.2 Register Overview

### Table 30: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x10001000 | UICR | UICR | User Information Configuration Registers |

### Table 31: Register Overview

| Register | Offset | Description |
|---|---|---|
| Registers | | |
| *CLENR0* | 0x000 | Length of code region 0 |
| *RBPCONF* | 0x004 | Read back protection configuration |
| *XTALFREQ* | 0x008 | Reset value for XTALFREQ in CLOCK, see CLOCK chapter |
| *FWID* | 0x010 | Firmware ID |
| *BOOTLOADERADDR* | 0x014 | Bootloader address |
| *NRFFW[0]* | 0x014 | Reserved for Nordic firmware design |
| *NRFFW[1]* | 0x018 | Reserved for Nordic firmware design |
| *NRFFW[2]* | 0x01C | Reserved for Nordic firmware design |
| *NRFFW[3]* | 0x020 | Reserved for Nordic firmware design |
| *NRFFW[4]* | 0x024 | Reserved for Nordic firmware design |
| *NRFFW[5]* | 0x028 | Reserved for Nordic firmware design |
| *NRFFW[6]* | 0x02C | Reserved for Nordic firmware design |
| *NRFFW[7]* | 0x030 | Reserved for Nordic firmware design |
| *NRFFW[8]* | 0x034 | Reserved for Nordic firmware design |
| *NRFFW[9]* | 0x038 | Reserved for Nordic firmware design |
| *NRFFW[10]* | 0x03C | Reserved for Nordic firmware design |
| *NRFFW[11]* | 0x040 | Reserved for Nordic firmware design |
| *NRFFW[12]* | 0x044 | Reserved for Nordic firmware design |
| *NRFFW[13]* | 0x048 | Reserved for Nordic firmware design |
| *NRFFW[14]* | 0x04C | Reserved for Nordic firmware design |
| *NRFHW[0]* | 0x050 | Reserved for Nordic hardware design |
| *NRFHW[1]* | 0x054 | Reserved for Nordic hardware design |
| *NRFHW[2]* | 0x058 | Reserved for Nordic hardware design |
| *NRFHW[3]* | 0x05C | Reserved for Nordic hardware design |
| *NRFHW[4]* | 0x060 | Reserved for Nordic hardware design |
| *NRFHW[5]* | 0x064 | Reserved for Nordic hardware design |
| *NRFHW[6]* | 0x068 | Reserved for Nordic hardware design |
| *NRFHW[7]* | 0x06C | Reserved for Nordic hardware design |
| *NRFHW[8]* | 0x070 | Reserved for Nordic hardware design |
| *NRFHW[9]* | 0x074 | Reserved for Nordic hardware design |
| *NRFHW[10]* | 0x078 | Reserved for Nordic hardware design |
| *NRFHW[11]* | 0x07C | Reserved for Nordic hardware design |
| *CUSTOMER[0]* | 0x080 | Reserved for customer |
| *CUSTOMER[1]* | 0x084 | Reserved for customer |

| Register | Offset | Description |
|---|---|---|
| CUSTOMER[2] | 0x088 | Reserved for customer |
| CUSTOMER[3] | 0x08C | Reserved for customer |
| CUSTOMER[4] | 0x090 | Reserved for customer |
| CUSTOMER[5] | 0x094 | Reserved for customer |
| CUSTOMER[6] | 0x098 | Reserved for customer |
| CUSTOMER[7] | 0x09C | Reserved for customer |
| CUSTOMER[8] | 0x0A0 | Reserved for customer |
| CUSTOMER[9] | 0x0A4 | Reserved for customer |
| CUSTOMER[10] | 0x0A8 | Reserved for customer |
| CUSTOMER[11] | 0x0AC | Reserved for customer |
| CUSTOMER[12] | 0x0B0 | Reserved for customer |
| CUSTOMER[13] | 0x0B4 | Reserved for customer |
| CUSTOMER[14] | 0x0B8 | Reserved for customer |
| CUSTOMER[15] | 0x0BC | Reserved for customer |
| CUSTOMER[16] | 0x0C0 | Reserved for customer |
| CUSTOMER[17] | 0x0C4 | Reserved for customer |
| CUSTOMER[18] | 0x0C8 | Reserved for customer |
| CUSTOMER[19] | 0x0CC | Reserved for customer |
| CUSTOMER[20] | 0x0D0 | Reserved for customer |
| CUSTOMER[21] | 0x0D4 | Reserved for customer |
| CUSTOMER[22] | 0x0D8 | Reserved for customer |
| CUSTOMER[23] | 0x0DC | Reserved for customer |
| CUSTOMER[24] | 0x0E0 | Reserved for customer |
| CUSTOMER[25] | 0x0E4 | Reserved for customer |
| CUSTOMER[26] | 0x0E8 | Reserved for customer |
| CUSTOMER[27] | 0x0EC | Reserved for customer |
| CUSTOMER[28] | 0x0F0 | Reserved for customer |
| CUSTOMER[29] | 0x0F4 | Reserved for customer |
| CUSTOMER[30] | 0x0F8 | Reserved for customer |
| CUSTOMER[31] | 0x0FC | Reserved for customer |

## 8.3 Register Details

### Table 32: CLENR0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | CLENR0 | | | Length of code region 0<br>Length of code region 0 in bytes. The value must be a multiple of "Code page size" bytes CODEPAGESIZE. N (max value) is (CODEPAGESIZE * ( CODESIZE - 1 )). This register can only be written if content is 0xFFFFFFFF. Value after mass erase of flash is 0xFFFFFFFF, this value is interpreted as 0. |

### Table 33: RBPCONF

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PR0 | | | Protect region 0. Enable or disable read-back protection of code region 0.<br>Will be ignored if pre-programmed factory Code is present on the chip. |
| | | | Disabled | 0xFF | Disable |
| | | | Enabled | 0x00 | Enable |
| B | RW | PALL | | | Protect all. Enable or disable read-back protection of all code in device. |
| | | | Disabled | 0xFF | Disable |
| | | | Enabled | 0x00 | Enable |

### Table 34: XTALFREQ

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | XTALFREQ | | | Reset value for XTALFREQ in CLOCK, see CLOCK chapter |
| | | | 16MHz | 0xFF | 16 MHz crystal is used |
| | | | 32MHz | 0x00 | 32 MHz crystal is used |

## Table 35: FWID

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | FWID | | | Firmware ID |

## Table 36: BOOTLOADERADDR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | BOOTLOADERADDR | | | Bootloader address |

## Table 37: NRFFW[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | NRFFW | | | Reserved for Nordic firmware design |

## Table 38: NRFHW[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | NRFHW | | | Reserved for Nordic hardware design |

## Table 39: CUSTOMER[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 | A | A | A | A | A | A | A | A |
| Reset | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | CUSTOMER | | | Reserved for customer |

# 9 Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) can protect the entire memory against readback and also protect parts of the memory area from accidental access by the CPU.

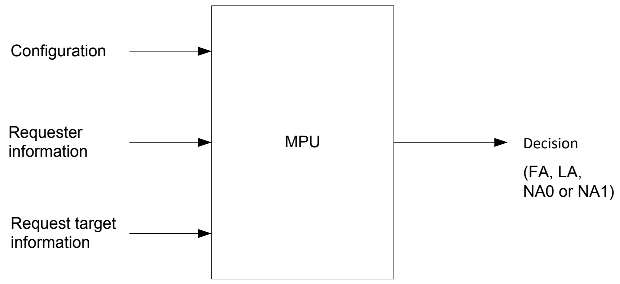



**Figure 4: Block diagram**

## 9.1 Functional description

Protect all (PALL) is configured by writing '0' to UICR.RBPCONF.PALL. When protect all is enabled, the debugger (SWD) will no longer have access to code region 0, code region 1, RAM or any peripherals except for the following:

- The NVMC peripheral.
- The RESET register in the POWER peripheral.
- The DISABLEINDEBUG register in the MPU peripheral.

Code memory, RAM, and peripherals can be divided into two regions: region 0 and region 1. Code memory regions are configured in the CLENR0 register in the User Information Configuration Register (UICR). When memory protection is enabled, these regions will be used by the Memory Protection Unit to enforce runtime protection and readback protection of resources classified as region 0.

Independent of protection settings, code region R0 (CR0) will always have full access to the system. The NVMC.ERASEPCR0 register, which is used to erase content from code region 0, can only be accessed from a program in code region 0.

Only the CPU can do fetches from code memory, and these will always be granted.

Except when generated by the SWD interface, accesses that are not granted by the MPU may result in a HardFault or RAM corruption.

Readback protection of code region 0 is enabled by writing '0' to UICR.RBPCONF.PR0. When enabled, only code running from code region 0 will be able to access the code in code region 0. Accesses generated by code running from code region 1 or from RAM, as well as accesses generated by the debugger (SWD), will not be granted when code region 0 is protected.

Independent of readback protection configuration of code region 0 the vector table, which is located between addresses 0x00000000 and 0x00000080, will not be protected by UICR.RBPCONF.PR0.

The main role for the two region memory protection system is to allow run time protection for SoftDevices installed on the IC.

### 9.1.1 Inputs

The MPU has three classes of inputs. These are:

- Configuration
  - Readback protection configuration from UICR and FICR.
- Information about requester

- Source of memory access request (SWD or CPU program).
- If the request source is a CPU program; region from which the program is running (region 0 or region 1).
- Types of access request (read or write).

- Target information
  - Memory category requested access to (code, RAM, or PER).
  - Memory region requested access to (region 0 or region 1).

### 9.1.2 Output

The MPU outputs the level of memory access that shall be given to a memory access request. The access levels the MPU can give are as follows:

- Full access (FA)
  - Full read write access to the requested memory.
- Limited access (LA)
  - Full read access.
  - No write access. Write will generate hard fault exception.
- No access 0 (NA0)
  - No read or write access.
  - Read will return 0.
  - Write will have no effect.
- No access 1 (NA1)
  - No read or write access.
  - Read or write will generate hard fault exception.

### 9.1.3 Output decision table

The output MPU access level based on the MPU inputs is given in the table below.

The given access level is dependent on settings in the Information Configuration Registers (ICRs). See the *UICR* and *FICR* chapters for more details.

#### Table 40: MPU output decision table based on the MPU inputs and the ICR configuration

| **Request source** | **UICR.RBPCONF.PALL (Readback protect entire code memory)** | **UICR.RBPCONF.PRO or FICR.PPFC (Readback protect code region 0)** | **Request target** Code R0 | Code R1 | RAM R0 | RAM R1 | PER R0 | PER R1 |
|---|---|---|---|---|---|---|---|---|
| SWD | 0xFF | 0xFF | FA | FA | FA | FA | FA | FA |
| | 0xFF | 0x00 | NA0 | FA | FA | FA | FA | FA |
| | 0x00 | X | NA0 | NA0 | NA0 | NA0 | NA0 | NA0 |
| Code R0 | X | X | FA | FA | FA | FA | FA | FA |
| Code R1 | X | 0xFF | LA | FA | LA | FA | LA | FA |
| | X | 0x00 | NA1 | FA | LA | FA | LA | FA |
| RAM R0/R1 | 0xFF | 0xFF | FA | FA | FA | FA | FA | FA |
| | 0xFF | 0x00 | NA1 | FA | FA | FA | FA | FA |
| | 0x00 | X | NA1 | NA1 | FA | FA | FA | FA |

**Key:**

- X: Don't care
- LA: limited access
- NA0: no access 0
- NA1: no access 1
- FA: full access

### 9.1.4 Exceptions from table

There are some exceptions from *Table 40: MPU output decision table based on the MPU inputs and the ICR configuration* on page 29. These exceptions are:

- The NVMC.ERASEALL and NVMC.ERASEUICR registers have conditional write access depending on the readback protection settings in the Information Configuration registers. These exceptions are described in the *NVMC* chapter.
- The NVMC.ERASEPCR0 register can only be accessed from a program in code region 0.
- The UICR.CLENR0 and the FICR. CLENR0 registers can only be modified when the register value equals the default value (0xFF). This is to avoid that the memory region limits are modified to bypass readback protection.

### 9.1.5 NVM protection blocks

The protection mechanism for NVM can be used to prevent erroneous application code from erasing or writing to protected blocks. Non-volatile memory can be protected from erases/writes depending on settings in the PROTENSET registers. One bit in a PROTENSET register represents one protected block. There are two PROTENSET registers of 32 bits which means there are 64 protectable blocks in total.

> **Note:** If an erase or write to a protected block is detected, the CPU will hard fault. If an ERASEALL operation is attempted from the CPU while any block is protected it will be blocked and the CPU will hard fault.

On reset, all the protection bits are cleared. To ensure safe operation, the first task after reset must be to set the protection bits. The only way of clearing protection bits is by resetting the device from any reset source.

The protection mechanism is turned off when in debug mode (a debugger is connected) and the DISABLEINDEBUG register is set to disable.

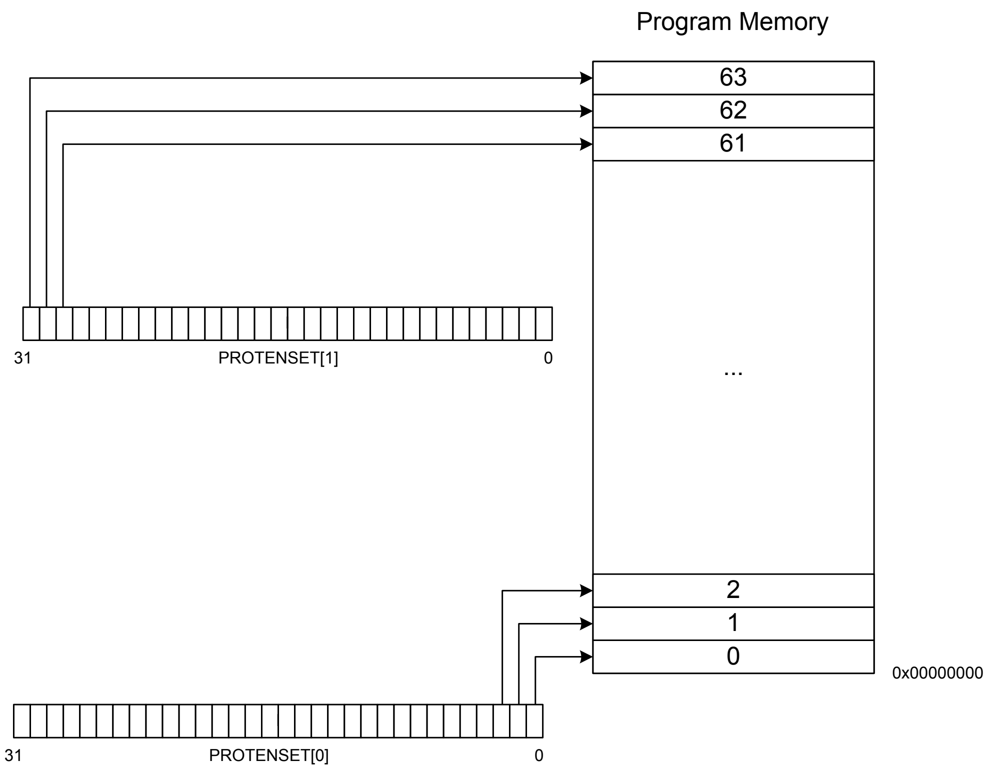



**Figure 5: Protected regions of program memory**

## 9.2 Register Overview

**Table 41: Instances**

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40000000 | MPU | MPU | Memory Protection Unit |

### Table 42: Register Overview

| Register | Offset | Description |
|---|---|---|
| Registers | | |
| *PERR0* | 0x528 | Definition of peripherals in memory region 0 |
| *RLENR0* | 0x52C | Length of RAM region 0 |
| *PROTENSET0* | 0x600 | Protection bit enable set register |
| *PROTENSET1* | 0x604 | Protection bit enable set register |
| *DISABLEINDEBUG* | 0x608 | Disable protection mechanism in debug mode |
| *PROTBLOCKSIZE* | 0x60C | Protection block size |

## 9.3 Register Details

### Table 43: PERR0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | U | T | | | | | | | | | | | S | R | Q | P | O | N | M | L | K | J | I | H | G | F | | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | POWER_CLOCK | | | Classify POWER and CLOCK, and all other peripherals with ID=0, as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| B | RW | RADIO | | | Classify RADIO as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| C | RW | UART0 | | | Classify UART0 as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| D | RW | SPI0_TWI0 | | | Classify SPI0 and TWI0 as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| E | RW | SPI1_TWI1 | | | Classify SPI1 and TWI1 as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| F | RW | GPIOTE | | | Classify GPIOTE as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| G | RW | ADC | | | Classify ADC as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| H | RW | TIMER0 | | | Classify TIMER0 as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| I | RW | TIMER1 | | | Classify TIMER1 as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| J | RW | TIMER2 | | 0 | Classify TIMER2 as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| K | RW | RTC0 | | | Classify RTC0 as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| L | RW | TEMP | | | Classify TEMP as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| M | RW | RNG | | | Classify RNG as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| N | RW | ECB | | | Classify ECB as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| O | RW | CCM_AAR | | | Classify CCM and ECB as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| P | RW | WDT | | | Classify WDT as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| Q | RW | RTC1 | | | Classify RTC1 as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| R | RW | QDEC | | | Classify QDEC as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| S | RW | LPCOMP | | | Classify LPCOMP as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |
| T | RW | NVMC | | | Classify NVMC as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 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 | U | T | | | | | | | | | | | S | R | Q | P | O | N | M | L | K | J | I | H | G | F | | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| U | RW | PPI | | | Classify PPI as region 0 or region 1 peripheral |
| | | | InRegion0 | 1 | Peripheral configured in region 0 |
| | | | InRegion1 | 0 | Peripheral configured in region 1 |

## Table 44: RLENR0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RLENR0 | | | This register specifies the size of RAM region 0 Given a base address for the RAM called RAMBA, RAM addresses < RAMBA + RLENR0 are classified as region 0 RAM and RAM addresses >= RAMBA + RLENR0 are classified as region 1 RAM. The address (RAMBA + RLENR0) has to be word-aligned. RAMBA and the total available RAM is defined in the product specification of the chip you are using. |

## Table 45: PROTENSET0

**Note:** Read: Read back value of protection bit {i}.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PROTREG0 | | | Write '1': Protection enable bit for region 0. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| B | RW | PROTREG1 | | | Write '1': Protection enable bit for region 1. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| C | RW | PROTREG2 | | | Write '1': Protection enable bit for region 2. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| D | RW | PROTREG3 | | | Write '1': Protection enable bit for region 3. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| E | RW | PROTREG4 | | | Write '1': Protection enable bit for region 4. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| F | RW | PROTREG5 | | | Write '1': Protection enable bit for region 5. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| G | RW | PROTREG6 | | | Write '1': Protection enable bit for region 6. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| H | RW | PROTREG7 | | | Write '1': Protection enable bit for region 7. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| I | RW | PROTREG8 | | | Write '1': Protection enable bit for region 8. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| J | RW | PROTREG9 | | | Write '1': Protection enable bit for region 9. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| K | RW | PROTREG10 | | | Write '1': Protection enable bit for region 10. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| L | RW | PROTREG11 | | | Write '1': Protection enable bit for region 11. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| M | RW | PROTREG12 | | | Write '1': Protection enable bit for region 12. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |

**Note:** Read: Read back value of protection bit {i}.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| N | RW | PROTREG13 | | | Write '1': Protection enable bit for region 13. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| O | RW | PROTREG14 | | | Write '1': Protection enable bit for region 14. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| P | RW | PROTREG15 | | | Write '1': Protection enable bit for region 15. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| Q | RW | PROTREG16 | | | Write '1': Protection enable bit for region 16. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| R | RW | PROTREG17 | | | Write '1': Protection enable bit for region 17. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| S | RW | PROTREG18 | | | Write '1': Protection enable bit for region 18. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| T | RW | PROTREG19 | | | Write '1': Protection enable bit for region 19. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| U | RW | PROTREG20 | | | Write '1': Protection enable bit for region 20. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| V | RW | PROTREG21 | | | Write '1': Protection enable bit for region 21. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| W | RW | PROTREG22 | | | Write '1': Protection enable bit for region 22. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| X | RW | PROTREG23 | | | Write '1': Protection enable bit for region 23. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| Y | RW | PROTREG24 | | | Write '1': Protection enable bit for region 24. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| Z | RW | PROTREG25 | | | Write '1': Protection enable bit for region 25. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AA | RW | PROTREG26 | | | Write '1': Protection enable bit for region 26. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AB | RW | PROTREG27 | | | Write '1': Protection enable bit for region 27. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AC | RW | PROTREG28 | | | Write '1': Protection enable bit for region 28. Write '0': no effect. |

**Note:** Read: Read back value of protection bit {i}.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AD | RW | PROTREG29 | | | Write '1': Protection enable bit for region 29. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AE | RW | PROTREG30 | | | Write '1': Protection enable bit for region 30. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AF | RW | PROTREG31 | | | Write '1': Protection enable bit for region 31. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |

## Table 46: PROTENSET1

**Note:** Read: Read back value of protection bit {i}.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PROTREG32 | | | Write '1': Protection enable bit for region 32. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| B | RW | PROTREG33 | | | Write '1': Protection enable bit for region 33. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| C | RW | PROTREG34 | | | Write '1': Protection enable bit for region 34. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| D | RW | PROTREG35 | | | Write '1': Protection enable bit for region 35. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| E | RW | PROTREG36 | | | Write '1': Protection enable bit for region 36. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| F | RW | PROTREG37 | | | Write '1': Protection enable bit for region 37. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| G | RW | PROTREG38 | | | Write '1': Protection enable bit for region 38. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| H | RW | PROTREG39 | | | Write '1': Protection enable bit for region 39. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| I | RW | PROTREG40 | | | Write '1': Protection enable bit for region 40. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| J | RW | PROTREG41 | | | Write '1': Protection enable bit for region 41. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| K | RW | PROTREG42 | | | Write '1': Protection enable bit for region 42. Write '0': no effect. |

**Note:** Read: Read back value of protection bit {i}.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| L | RW | PROTREG43 | | | Write '1': Protection enable bit for region 43. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| M | RW | PROTREG44 | | | Write '1': Protection enable bit for region 44. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| N | RW | PROTREG45 | | | Write '1': Protection enable bit for region 45. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| O | RW | PROTREG46 | | | Write '1': Protection enable bit for region 46. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| P | RW | PROTREG47 | | | Write '1': Protection enable bit for region 47. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| Q | RW | PROTREG48 | | | Write '1': Protection enable bit for region 48. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| R | RW | PROTREG49 | | | Write '1': Protection enable bit for region 49. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| S | RW | PROTREG50 | | | Write '1': Protection enable bit for region 50. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| T | RW | PROTREG51 | | | Write '1': Protection enable bit for region 51. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| U | RW | PROTREG52 | | | Write '1': Protection enable bit for region 52. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| V | RW | PROTREG53 | | | Write '1': Protection enable bit for region 53. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| W | RW | PROTREG54 | | | Write '1': Protection enable bit for region 54. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| X | RW | PROTREG55 | | | Write '1': Protection enable bit for region 55. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| Y | RW | PROTREG56 | | | Write '1': Protection enable bit for region 56. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| Z | RW | PROTREG57 | | | Write '1': Protection enable bit for region 57. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AA | RW | PROTREG58 | | | |

**Note:** Read: Read back value of protection bit {i}.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | | | Write '1': Protection enable bit for region 58. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AB | RW | PROTREG59 | | | Write '1': Protection enable bit for region 59. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AC | RW | PROTREG60 | | | Write '1': Protection enable bit for region 60. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AD | RW | PROTREG61 | | | Write '1': Protection enable bit for region 61. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AE | RW | PROTREG62 | | | Write '1': Protection enable bit for region 62. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |
| AF | RW | PROTREG63 | | | Write '1': Protection enable bit for region 63. Write '0': no effect. |
| | | | Disabled | 0 | Read: protection disabled |
| | | | Enabled | 1 | Read: protection enabled |
| | | | Set | 1 | Write: enables protection |

## Table 47: DISABLEINDEBUG

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | DISABLEINDEBUG | | | Disable the protection mechanism for NVM regions while in debug mode. This register will only disable the protection mechanism if the device is in debug mode. |
| | | | Disabled | 1 | Disable in debug |
| | | | Enabled | 0 | Enable in debug |

## Table 48: PROTBLOCKSIZE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PROTBLOCKSIZE | | | Protection block size |
| | | | 4k | 0 | 4 kByte protection block size |

# 10 Peripheral interface

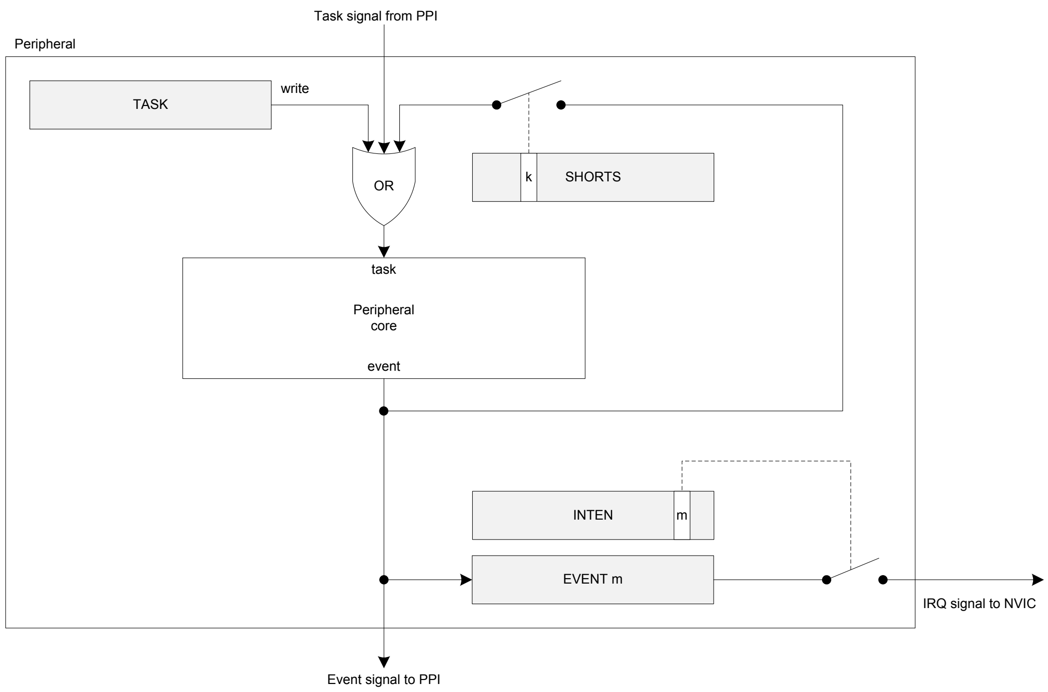



**Figure 6: Tasks, events, shortcuts, and interrupts**

## 10.1 Functional description

All peripherals can be accessed through the standard ARM® Cortex Advanced Peripheral Bus (APB) or AMBA High-performance Bus (AHB) registers as well as through task, event, and interrupt registers.

### 10.1.1 Peripheral ID

Every peripheral is assigned a fixed block of 0x1000 bytes, which is equal to 1024 x 32 bit registers. This pattern is applied to all peripherals located on the APB bus and on the AHB bus. See *Instantiation* on page 17 for more information about which peripherals are available and where they are located in the address map.

For peripherals on the APB bus there is a direct relationship between its ID and its base address. A peripheral with base address 0x40000000 is therefore assigned ID=0, and a peripheral with base address 0x40001000 is assigned ID=1. The 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:

- Peripherals do not share any registers or common resources, but the total number of registers available for each peripheral is reduced compared to a peripheral that has a dedicated ID.
- Peripherals share some registers or other common resources.
- Mutually exclusive. Only one of the peripherals can be used at a time.
- Both peripherals are optional in the series, and only one of them is instantiated in any given chip.
- Switching from one peripheral to another must follow a specific pattern (disable the first, then enable the second peripheral).

When switching between two mutually exclusive peripherals that share the same ID the user should do the following to prevent unwanted behaviour:

- Remove any PPI connections set up for the peripheral that is being disabled

- Clear all bits in the INTEN register, i.e. INTENCLR = 0xFFFFFFFF.
- Explicitly configure the peripheral that you enable and do not rely on configuration values that may be inherited from the peripheral that was disabled.

### 10.1.2 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 a dedicated SET and CLR register in that 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 a '1' to a bit in the SET or CLR register will set or clear the same bit in the main register respectively. Writing a '0' to a bit in the SET or CLR register has no effect. Reading the SET or CLR registers returns the value of the main register.

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

### 10.1.3 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 a '1' to the task register or when the peripheral itself, or another peripheral, toggles the corresponding task signal. *Figure 6: Tasks, events, shortcuts, and interrupts* on page 37

### 10.1.4 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 with each event having a separate register in that peripheral's event register group.

An event is generated when the peripheral itself toggles the corresponding event signal, whereupon the event register is updated to reflect that the event has been generated. See *Figure 6: Tasks, events, shortcuts, and interrupts* on page 37. An event register is only cleared when firmware writes a '0' to it.

Events can be generated by the peripheral even when the event register is set to '1'.

### 10.1.5 Shortcuts

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

Using a shortcut is the equivalent to making the same connection outside the peripheral and through the PPI. However, the propagation delay through the shortcut is usually shorter than the propagation delay through the PPI.

Shortcuts are predefined, which means 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.

### 10.1.6 Interrupts

An interrupt is an exception that is generated by an event and can interrupt the program flow of the CPU. All peripherals on the APB bus support interrupts. 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 Vector Interrupt Controller (NVIC).

Using the INTEN, INTENSET and INTENCLR registers, you can configure every event in a peripheral to generate that peripheral's interrupt. You can enable multiple events to generate interrupts simultaneously. To resolve the correct interrupt's source, firmware can query the event registers found in the event group in the peripherals register map.

Some peripherals implement only INTENSET and INTENCLR, the INTEN register is not available on those peripherals. Refer to the individual chapters for details. In all cases, however, 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 correct bit position can be derived from the event's address. The event on address 0x100 is associated with bit 0 in the INTEN register, the event at address 0x104 is associated with bit 1, and so on. The event at address 0x17C is identified with bit 31 in the INTEN register. This pattern effectively limits the maximum number of events in a peripheral to 32.

The relationship between tasks, events, shortcuts, and interrupts is shown in *Figure 6: Tasks, events, shortcuts, and interrupts* on page 37.

# 11 Debugger Interface (DIF)

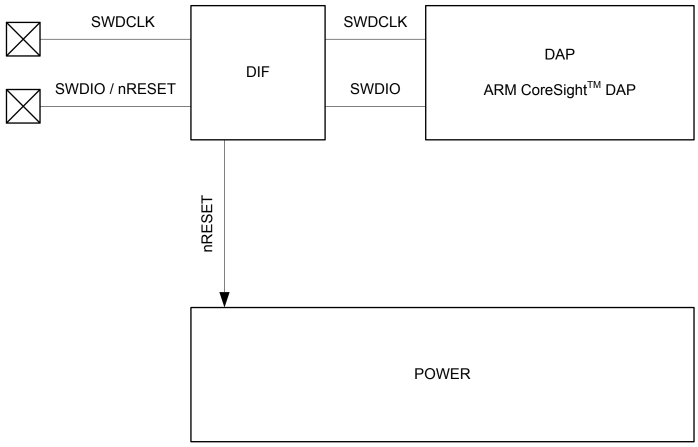



**Figure 7: Debugger interface**

## 11.1 Functional description

nRF51 devices support the Serial wire Debug (SWD) interface from ARM. The interface has two lines; SWDCLK and SWDIO. SWDIO and nRESET share the same physical pin. The Debugger Interface (DIF) module is responsible for handling the resource sharing between SWD traffic and reset functionality. The SWDCLK pin has an internal pull down resistor and the SWDIO/nRESET pin has an internal pull up resistor.

### 11.1.1 Normal mode

The DIF module will be in normal mode after power on reset. In this mode the SWDIO/nRESET pin acts as a normal active low reset pin.

To guarantee that the device remains in normal mode, the SWDCLK line must be held low, that is, '0', at all times. Failing to do so may result in the DIF entering into an unknown state and may lead to undesirable behavior and power consumption.

### 11.1.2 Debug interface mode

Debug interface mode is initiated by clocking one clock cycle on SWDCLK with SWDIO=1. Due to delays caused by starting up the DAP's power domain, a minimum of 150 clock cycles must be clocked at a speed of minimum 125 kHz on SWDCLK with SWDIO=1 to guarantee that the DAP is able to capture a minimum of 50 clock cycles.

If the device is in System OFF mode, see *Power management (POWER)* on page 42 for more information about System OFF mode, entering into debug interface mode will generate a wakeup.

In debug interface mode, the SWDIO/nRESET pin will be used as SWDIO. The pin reset mechanism will therefore be disabled as long as the device is in debug interface mode.

In debug interface mode, System OFF will be emulated to facilitate debugging of the device while in System OFF. Power numbers will naturally be higher in emulated System OFF compared to normal System OFF. See *Emulated System OFF mode* on page 44 for more information.

### 11.1.3 Resuming normal mode

Normal mode can always be resumed by performing a "hard-reset" through the SWD interface:

1. Enter debug interface mode.
2. Enable reset through the RESET register in the POWER peripheral.

3. Hold the `SWDCLK` and `SWDIO/nRESET` line low for a minimum of 100 µs.

You can also generate a "hard-reset" by performing a power on reset, or a brown-out reset.

# 12 Power management (POWER)

## 12.1 Functional description

Power management architecture gives you unique flexibility through orthogonal power control of all system blocks on the devices.

### 12.1.1 Power supply

The following power supply alternatives are supported:

- Internal DC/DC converter setup
- Internal LDO setup
- Low Voltage mode setup

### 12.1.2 Internal LDO setup

The internal DC/DC converter can be bypassed if it is not going to be used. When the DC/DC converter is bypassed, only the internal LDO is active as illustrated in *Figure 8: LDO regulator only* on page 42. The internal LDO will then generate the system power directly from the supply voltage VDD. It is recommended that the DC/DC converter is disabled in this setup.

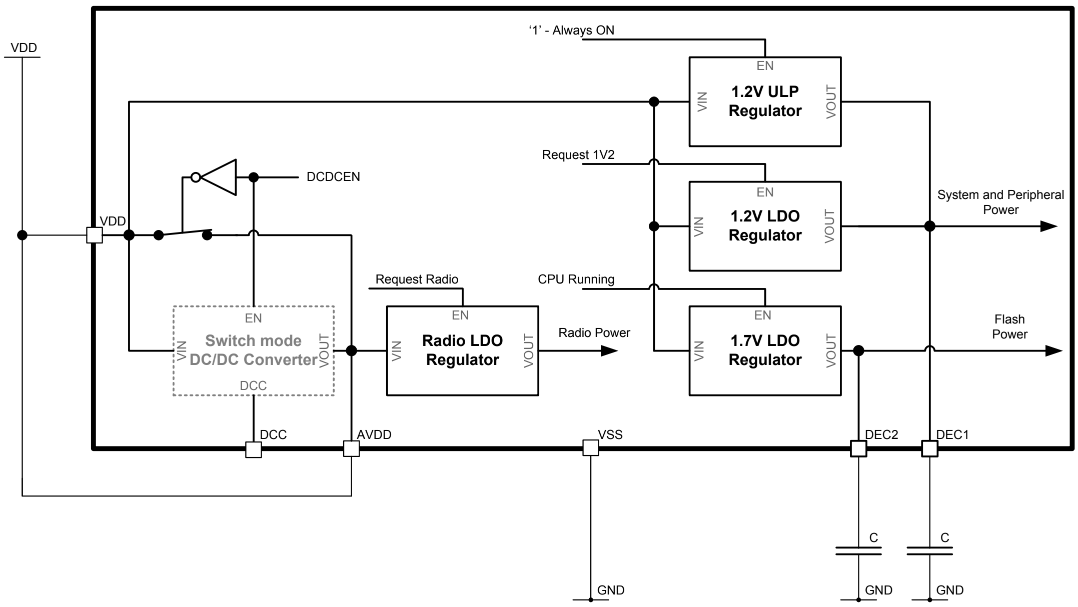



**Figure 8: LDO regulator only**

### 12.1.3 DC/DC converter setup

Selected devices have a Buck type DC/DC converter that steps down the supply voltage VDD. The resulting voltage is then used by an internal LDO that supplies the radio with power.

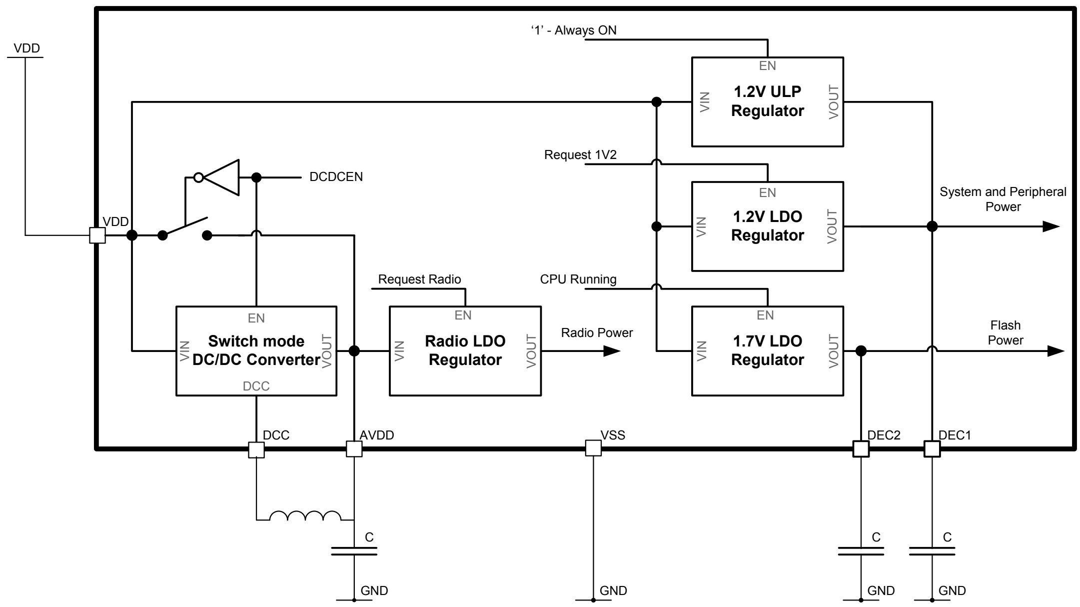



**Figure 9: DC/DC converter**

The DC/DC converter requires an external LC filter and is enabled through the *DCDCEN* register. See the reference circuitry chapter in the product specification for more information about component values.

The DC/DC converter only reduces the power consumption used by the radio, it does not affect the power used by the Flash, System, and Peripheral.

Enabling the DC/DC converter will not turn it on, but set it in a state where it automatically gets turned on when the radio is enabled and goes off again when the radio gets disabled. This is done to avoid wasting power running the DC/DC in between the radio events where current consumption is too low.

### DC/DC efficiency

The conversion factor ($F_{DCDC}$) is the ratio between the power used by the radio and the DC/DC converter when the DC/DC is active ($I_{DD,DCDC}$) and the power used by the radio when the DC/DC is disabled ($I_{DD}$). As shown in below:

$$I_{DD,DCDC} = F_{DCDC} * I_{DD}$$

The conversion factor ($F_{DCDC}$) depends on two parameters:

- Supply voltage (VDD).
- Current consumption used by the radio ($I_{DD}$).

The conversion factor ($F_{DCDC}$) will increase with decreasing supply voltage (VDD) if the current drawn through the DC/DC converter (Radio power) is kept constant. The conversion factor ($F_{DCDC}$) also increases with decreasing current consumption ($I_{DD}$), for a given voltage (VDD).

If we look at these two parameters in combination we will find a limit where the DC/DC converter no longer reduces the power consumption (i.e. $F_{DCDC} > 1$).

For data on the DC/DC performance see product specification.

## 12.1.4 Low voltage mode setup

If you have a stable, low voltage available for the nRF51 device, it is possible to configure the device in low voltage mode as illustrated in *Figure 10: Low voltage mode* on page 44. In this mode the internal LDO is bypassed and the system is powered directly from the supply voltage VDD. See the product specification for more information about which voltage levels are supported in low voltage mode. In low voltage mode, the DC/DC converter must be disabled. Additional requirements may apply to the accuracy and stability of the supply voltage in low voltage mode. See the product specification for more information.

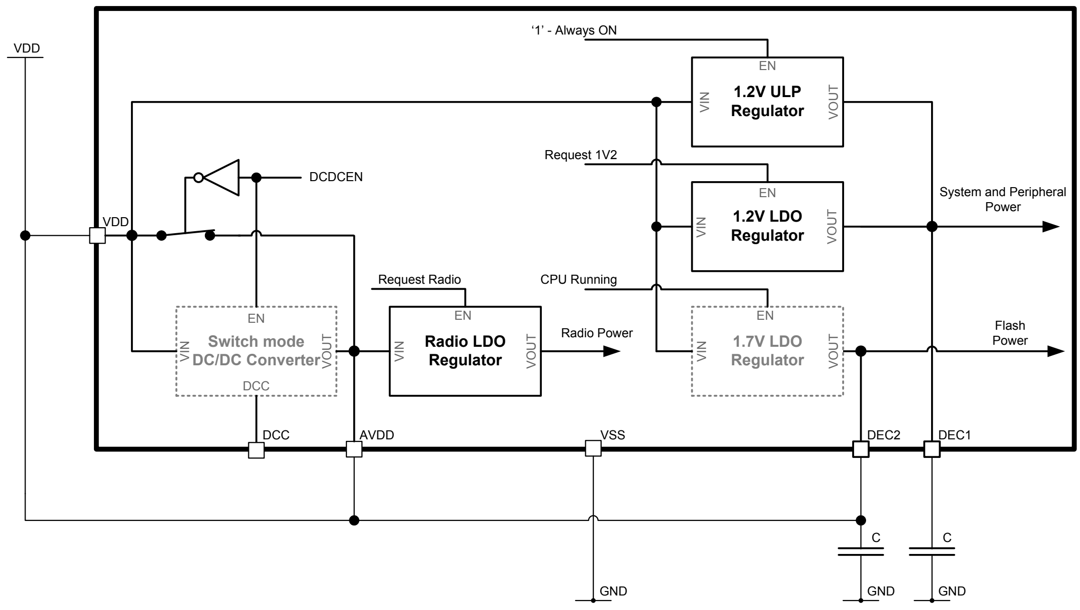



**Figure 10: Low voltage mode**

## 12.1.5 System OFF mode

System OFF is the deepest power saving mode the system can enter. In this mode, the system's core functionality is powered down and all ongoing tasks are terminated. The only mechanism that is functional and responsive in this mode is the reset and the wakeup mechanism.

One or more blocks of RAM can be retained in System OFF mode depending on the settings in the RAMON (and RAMONB, if provided) register(s).

RAMON and RAMONB are retained registers, see *Reset behaviour*. Note that these registers are usually overwritten by the startup code provided with the nRF application examples.

The system can be woken up from System OFF mode either from the DETECT signal (when active) generated by the *GPIO* peripheral, by the ANADETECT signal (when active) generated by the *LPCOMP* module, or from a reset. When the system wakes up from OFF mode, a system reset is performed.

Before entering system OFF mode the user must make sure that all on-going EasyDMA transactions have been completed. This is usually accomplished by making sure that the EasyDMA enabled peripheral is not active when entering system OFF. See documentation of these peripherals for more information.

## 12.1.6 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 *DIF* chapter for more information. Required resources needed for debugging include the following key components: DIF, CLOCK, POWER, NVMC, MPU, CPU, CODE, and RAM. Since the CPU is kept on in emulated System OFF mode, it is recommended to add an infinite loop directly after entering System OFF, to prevent the CPU from executing code that normally should not be executed.

## 12.1.7 System ON mode

System ON mode is a fully operational mode, where the CPU and all peripherals are brought into a state where they are functional.

In System ON mode the CPU can either be active or sleeping. The CPU enters sleep by executing the WFI or WFE instruction found in the CPU's instruction set. In WFI sleep the CPU will wake up as a result of an interrupt request if the associated interrupt is enabled in the NVIC. In WFE sleep the CPU will wake up as a result of an interrupt request regardless of the associated interrupt being enabled in the NVIC or not.

The system implements mechanisms to automatically switch on and off the appropriate power sources depending on how many peripherals are active, and how much power is needed at any given time. The power requirement of a peripheral is directly related to its activity level. The activity level is usually raised and lowered when specific tasks are triggered or events generated, see individual chapters describing the different peripherals for more information on how to optimize power consumption in System ON mode.

#### Sub power modes

During CPU sleep, in System ON mode, the system can reside in one of the following two sub power modes:

- Constant latency
- Low power

In constant latency mode (for more information, see the device specific product specification) the CPU wakeup latency and the PPI task response will be constant and kept at a minimum. This is secured by forcing a set of base resources on while in sleep, see the device specific product specification for more information about which resources are forced on. The advantage of having a constant and predictable latency will be at the cost of having increased power consumption. The constant latency mode is selected by triggering the CONSTLAT task.

In low power mode the automatic power management system, described in *System ON mode* on page 44, will be the most efficient and save the most power. The advantage of having low power will be at the cost of having varying CPU wakeup latency and PPI task response. The low power mode is selected by triggering the LOWPWR task.

When the system enters ON mode, it will, by default, reside in the low power sub-power mode.

### 12.1.8 Power supply supervisor

The power supply supervisor initializes the system at power-on and provides an early warning of impending power failure. In addition the power supply supervisor puts the system in a reset state if the supply voltage is too low for safe operation (brown-out). The power supply supervisor is illustrated in *Figure 11: Power supply supervisor* on page 45.

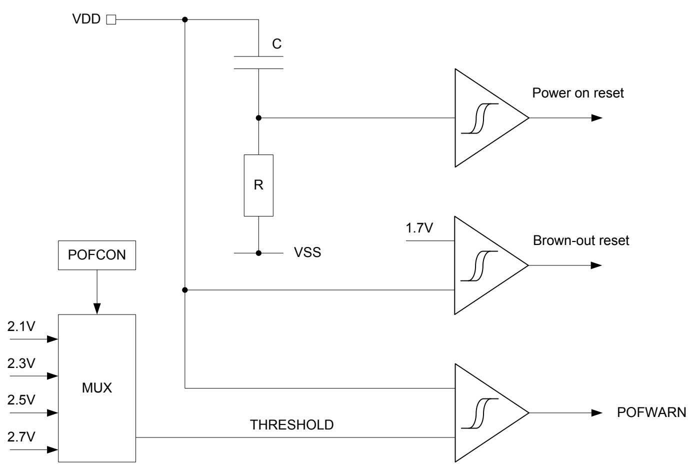



**Figure 11: Power supply supervisor**

### 12.1.9 Power-fail comparator

The power-fail comparator provides the CPU with an early warning of impending power failure. It will not reset the system, but give the CPU time to prepare for an orderly power-down. It also provides hardware protection of data stored in program memory by preventing write instructions from being executed. More information about this mechanism can be found in the *NVMC* chapter.

The comparator features a hysteresis of $V_{HYST}$ (refer to the Product Specification for the exact value), as illustrated in *Figure 12: Power failure comparator (BOR = Brown-out reset)* on page 46. The threshold $V_{POF}$ is set in the POFCON register.

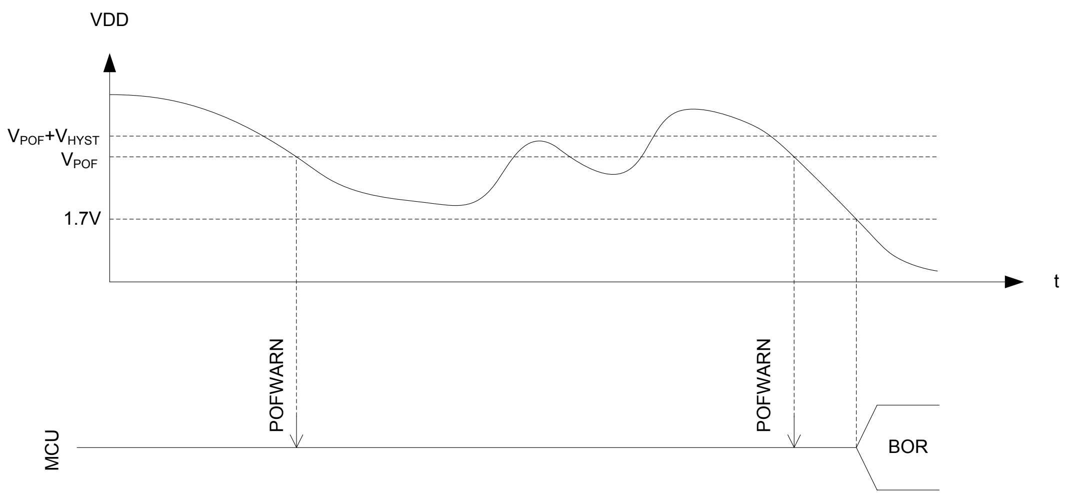



**Figure 12: Power failure comparator (BOR = Brown-out reset)**

To save power the power-fail comparator is not active in System OFF or in System ON when HFLKC is not running.

### 12.1.10 RAM blocks

Each of the available RAM blocks, which each may contain multiple RAM sections, can power up and down independently in both System ON and System OFF mode. See *Memory* chapter for more information about RAM blocks and sections.

### 12.1.11 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.

### 12.1.12 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, see the device specific product specification for more information.

### 12.1.13 Pin reset

A pin reset is generated when the physical reset pin on the device is asserted.

Since the debugger interface uses the same pin as the pin reset mechanism, a pin reset will not be available when the device is in debug interface mode unless explicitly enabled in the RESET register.

### 12.1.14 Wakeup from OFF mode reset

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

The DAP is not reset following a wake up from OFF mode if the device is in debug interface mode, see *DIF* chapter for more information.

### 12.1.15 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. RAM can be retained per setting in RAMON register.

### 12.1.16 Watchdog reset

A Watchdog reset is generated when the watchdog times out. See *WDT*

### 12.1.17 Brown-out reset

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

### 12.1.18 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.

### 12.1.19 Reset behavior

| Reset source | Reset target | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | CPU | Peripherals | GPIO | Debug[a] | RAM | WDT | Retained registers | RESETREAS |
| CPU lockup [2] | x | x | x | | | | | |
| Soft reset | x | x | x | | x | | | |
| Wakeup from System OFF mode reset | x | x | | x [3] | x | | | |
| Watchdog reset [4] | x | 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 | x |

**Note:** RAM content is never reset or cleared, but may loose content in the cases marked in the table above. RAM content can be retained over System Off or Soft reset depending on the setting of the RAMON registers.

## 12.2 Register Overview

### Table 49: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40000000 | POWER | POWER | Power control |

### Table 50: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *CONSTLAT* | 0x078 | Enable constant latency mode |
| *LOWPWR* | 0x07C | Enable low power mode (variable latency) |
| Events | | |
| *POFWARN* | 0x108 | Power failure warning |
| Registers | | |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *RESETREAS* | 0x400 | Reset reason |
| *RAMSTATUS* | 0x428 | RAM status register |
| *SYSTEMOFF* | 0x500 | System OFF register |
| *POFCON* | 0x510 | Power failure comparator configuration |
| *GPREGRET* | 0x51C | General purpose retention register |
| *RAMON* | 0x524 | RAM on/off register (this register is retained) |
| *RESET* | 0x544 | Reset configuration register |
| *RAMONB* | 0x554 | RAM on/off register (this register is retained) |
| *DCDCEN* | 0x578 | DC/DC enable register |

[a] See DIF chapter for more information about the different debug components in the system.
[2] Reset from CPU lockup is disabled if the device is in debug interface mode. CPU lockup is not possible in System OFF.
[3] The Debug components will not be reset if the device is in debug interface mode.
[4] Watchdog reset is not available in System OFF.

## 12.3 Register Details

### Table 51: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | POFWARN | | | Write '1' to Enable interrupt on *POFWARN* event. |
| | | | Enabled | 1 | Enable |

### Table 52: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | POFWARN | | | Write '1' to Clear interrupt on *POFWARN* event. |
| | | | Disabled | 1 | Disable |

### Table 53: RESETREAS

**Note:** 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 brown out 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RESETPIN | | | Reset from pin-reset detected |
| | | | NotDetected | 0 | Not detected |
| | | | Detected | 1 | Detected |
| B | RW | DOG | | | Reset from watchdog detected |
| | | | NotDetected | 0 | Not detected |
| | | | Detected | 1 | Detected |
| C | RW | SREQ | | | Reset from AIRCR.SYSRESETREQ detected |
| | | | NotDetected | 0 | Not detected |
| | | | Detected | 1 | Detected |
| D | RW | LOCKUP | | | Reset from CPU lock-up detected |
| | | | NotDetected | 0 | Not detected |
| | | | Detected | 1 | Detected |
| E | RW | OFF | | | Reset due to wake up from system OFF mode when wakeup is triggered from DETECT signal from GPIO |
| | | | NotDetected | 0 | Not detected |
| | | | Detected | 1 | Detected |
| F | RW | LPCOMP | | | Reset due to wake up from system OFF mode when wakeup is triggered from ANADETECT signal from LPCOMP |
| | | | NotDetected | 0 | Not detected |
| | | | Detected | 1 | Detected |
| G | RW | DIF | | | Reset due to wake up from system OFF mode when wakeup is triggered from entering into debug interface mode |
| | | | NotDetected | 0 | Not detected |
| | | | Detected | 1 | Detected |

### Table 54: RAMSTATUS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RAMBLOCK0 | | | RAM block 0 is on or off/powering up |
| | | | Off | 0 | Off |
| | | | On | 1 | On |
| B | R | RAMBLOCK1 | | | RAM block 1 is on or off/powering up |
| | | | Off | 0 | Off |
| | | | On | 1 | On |
| C | R | RAMBLOCK2 | | | RAM block 2 is on or off/powering up |
| | | | Off | 0 | Off |
| | | | On | 1 | On |
| D | R | RAMBLOCK3 | | | RAM block 3 is on or off/powering up |
| | | | Off | 0 | Off |
| | | | On | 1 | On |

### Table 55: SYSTEMOFF

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | W | SYSTEMOFF | | | Enable system OFF mode |
| | | | Enter | 1 | Enable system OFF mode |

### Table 56: POFCON

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | POF | | | Enable or disable power failure comparator |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | THRESHOLD | | | Power failure comparator threshold setting |
| | | | V21 | 0 | Set threshold to 2.1 V |
| | | | V23 | 1 | Set threshold to 2.3 V |
| | | | V25 | 2 | Set threshold to 2.5 V |
| | | | V27 | 3 | Set threshold to 2.7 V |

### Table 57: GPREGRET

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | GPREGRET | | | General purpose retention register<br>This register is a retained register |

### Table 58: RAMON

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | ONRAM0 | | | Keep RAM block 0 on or off in system ON Mode |
| | | | RAM0Off | 0 | Off |
| | | | RAM0On | 1 | On |
| B | RW | ONRAM1 | | | Keep RAM block 1 on or off in system ON Mode |
| | | | RAM1Off | 0 | Off |
| | | | RAM1On | 1 | On |
| C | RW | OFFRAM0 | | | Keep retention on RAM block 0 when RAM block is switched off |
| | | | RAM0Off | 0 | Off |
| | | | RAM0On | 1 | On |
| D | RW | OFFRAM1 | | | Keep retention on RAM block 1 when RAM block is switched off |
| | | | RAM1Off | 0 | Off |
| | | | RAM1On | 1 | On |

### Table 59: RESET

This register is a retained 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RESET | | | Enable or disable pin reset in debug interface mode, see the DIF peripheral chapter |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 60: RAMONB

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | ONRAM2 | | | Keep RAM block 2 on or off in system ON Mode |
| | | | RAM2Off | 0 | Off |
| | | | RAM2On | 1 | On |
| B | RW | ONRAM3 | | | Keep RAM block 3 on or off in system ON Mode |
| | | | RAM3Off | 0 | Off |
| | | | RAM3On | 1 | On |
| C | RW | OFFRAM2 | | | Keep retention on RAM block 2 when RAM block is switched off |
| | | | RAM2Off | 0 | Off |
| | | | RAM2On | 1 | On |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| D | RW | OFFRAM3 | | | Keep retention on RAM block 3 when RAM block is switched off |
| | | | RAM3Off | 0 | Off |
| | | | RAM3On | 1 | On |

## Table 61: DCDCEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DCDCEN | | | Enable or disable DC/DC converter |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

# 13 Clock management (CLOCK)

## 13.1 Functional description

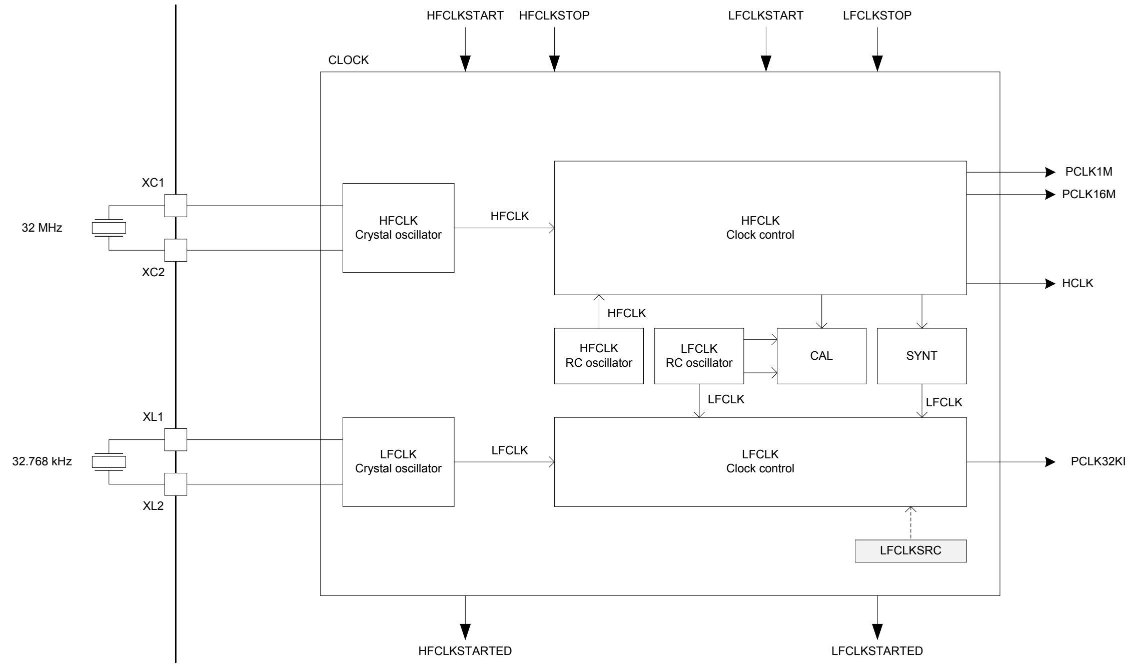



**Figure 13: Clock control**

### 13.1.1 HFCLK clock controller

As illustrated in *Figure 13: Clock control* on page 51 the system supports the following high frequency clock sources:

- HFCLK crystal oscillator : 16 or 32 MHz crystal oscillator
- HFCLK RC oscillator : 16 MHz RC oscillator

The system high frequency clock (HFCLK) is derived from one of the above clock sources.

The HFCLK crystal oscillators require an external AT-cut quartz crystal to be connected to the **XC1** and **XC2** pins in parallel resonant mode. If a 32 MHz crystal is used the XTALFREQ register must be configured accordingly.

The HFCLK clock controller provides the following clocks to the system derived from HFCLK:

- HCLK: 16 MHz high frequency clock for the CPU and the system as a whole.
- PCLK1M: 1 MHz peripheral clock.
- PCLK16M: 16 MHz peripheral clock.

These clocks are only available when the system is in ON mode.

When the system enters ON mode, HFCLK RC oscillator will start up automatically to provide the required clocks for the system.

The HFCLK crystal oscillator is started by triggering the HFCLKSTART task and stopped using the HFCLKSTOP task. A HFCLKSTARTED event will be generated when the selected HFCLK crystal oscillator has started. The start-up times of the HFCLK crystal oscillators are described in the device specific product specification.

A HFCLKSTOP task will stop the HFCLK oscillator. However the HFCLKSTOP task can only be sent after the STATE field in the HFCLKSTAT register indicates a 'HFCLK running' state.

The HFCLK RC oscillator is automatically switched off when the HFCLK crystal oscillators is running; it will be switched back on automatically when the HFCLK crystal oscillator is stopped.

If the system does not require any of the clocks provided by the HFCLK clock controller, the HFCLK controller may enter a power saving mode automatically and switch off the selected clock source. This occurs if all peripherals that require either PCLK1M, PCLK16M are appropriately stopped or disabled, and the CPU is sleeping and thereby no longer requesting HCLK.

When one or more of the clocks PCLK1M, PCLK16M or HFCLK are requested again, the HFCLK clock controller will resume normal operation mode. There will be transition time from power saving mode to normal operation mode that may be different depending on the HFCLK source, see product specification for more information.

To use the RADIO and the calibration mechanism associated with the 32.768 kHz RC oscillator, the HFCLK crystal oscillator must be running.

The HFCLK crystal oscillators utilize amplitude regulated architecture to achieve low current consumption and fast start-up. The HFCLK crystal oscillators are also designed to work with one of the following alternative external sources:

- A 16 MHz rail-to-rail clock signal applied to the XC1 pin. The XC2 pin shall then be left unconnected.
- A 16 MHz low swing clock signal applied to the XC1 pin. The XC2 pin shall then be left unconnected.

## 13.1.2 LFCLK clock controller

As illustrated in *Figure 13: Clock control* on page 51 the system supports the following low frequency clock sources:

- LFCLK crystal oscillator: 32.768 kHz crystal oscillator
- LFCLK RC oscillator: 32.768 kHz RC oscillator
- LFCLK synthesizer: 32.768 kHz synthesized from HFCLK
- LFCLK external clock source

The 32.768 kHz crystal oscillator requires an external AT-cut quartz crystal to be connected to the XL1 and XL2 pins in parallel resonant mode. The **XL1** and **XL2** share pins with the GPIO.

> **Note:** GPIOs that share pins with XL1 and XL2 differ from device to device. For more information, see the device specific product specification.

The LFCLK clock controller provides the following clocks to the system derived from LFCLK:

- PCLK32KI: 32.768 kHz low frequency clock for peripherals

The LFCLK clock controller and all of the LFCLK clock sources are switched off by default when the system is propagated from OFF to ON mode.

The LFCLK clock is started by first selecting the preferred clock source in the LFCLKSRC register and then triggering the LFCLKSTART task. If the selected clock source cannot be started immediately the 32.768 kHz RC oscillator will start automatically and generate the LFCLK until the selected clock source is available.

The LFCLK clock is stopped by triggering the LFCLKSTOP task. The LFCLKSRC register can only be modified when the LFCLK is not running.

A LFCLKSTARTED event will be generated when the selected LFCLK crystal oscillator has started. The start-up times of the LFCLK crystal oscillators are described in the device specific product specification.

A LFCLKSTOP task will stop the LFCLK oscillator. However, the LFCLKSTOP task can only be triggered after the STATE field in the LFCLKSTAT register indicates a 'LFCLK running' state.

The 32.768 kHz crystal oscillator utilizes an amplitude regulated architecture to achieve low current consumption and fast start-up.

The 32.768 kHz crystal oscillator is also designed to work with one of the following alternative external sources:

- A low swing clock signal applied to the **XL1** pin. The **XL2** pin shall then be left unconnected.

The synthesized 32.768 kHz clock depends on the HFCLK to run. If 250 ppm accuracy is required for the LFCLK running off the synthesized 32.768 kHz clock, the HFCLK must be generated from the HFCLK crystal oscillator.

## 13.1.3 Calibrating the 32.768 kHz RC oscillator

After the 32.768 kHz RC oscillator is started and running, it can be calibrated by triggering the CAL task. The 32.768 kHz RC oscillator will then temporarily request the HFCLK to calibrate itself against. A DONE event will be generated when calibration has finished. The calibration mechanism will only work as long as HFCLK is generated from the HFCLK crystal oscillator, it is therefore necessary to explicitly start this crystal oscillator before calibration can be started, see HFCLKSTART task. See product specification for recommendations on calibration intervals and crystal accuracy.

### 13.1.4 Calibration timer

The calibration timer can be used to time the calibration interval of the 32.768 kHz RC oscillator. The calibration timer is started by triggering the CTSTART task and stopped by triggering the CTSTOP task. The calibration timer will always start counting down from the value specified in CTIV and generate a CTTO timeout event when it reaches 0. The Calibration timer will stop by itself when it reaches 0.

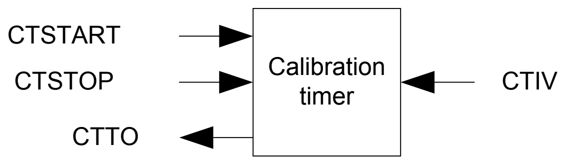



**Figure 14: Calibration timer**

Due to limitations in the calibration timer, only one task related to calibration, that is, CAL, CTSTART and CTSTOP, can be triggered for every period of LFCLK.

## 13.2 Register Overview

**Table 62: Instances**

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40000000 | CLOCK | CLOCK | Clock control |

**Table 63: Register Overview**

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *HFCLKSTART* | 0x000 | Start HFCLK crystal oscillator |
| *HFCLKSTOP* | 0x004 | Stop HFCLK crystal oscillator |
| *LFCLKSTART* | 0x008 | Start LFCLK source |
| *LFCLKSTOP* | 0x00C | Stop LFCLK source |
| *CAL* | 0x010 | Start calibration of LFCLK RC oscillator |
| *CTSTART* | 0x014 | Start calibration timer |
| *CTSTOP* | 0x018 | Stop calibration timer |
| Events | | |
| *HFCLKSTARTED* | 0x100 | HFCLK oscillator started |
| *LFCLKSTARTED* | 0x104 | LFCLK started |
| *DONE* | 0x10C | Calibration of LFCLK RC oscillator complete event |
| *CTTO* | 0x110 | Calibration timer timeout |
| Registers | | |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *HFCLKRUN* | 0x408 | Status indicating that *HFCLKSTART* task has been triggered |
| *HFCLKSTAT* | 0x40C | Which HFCLK source is running |
| *LFCLKRUN* | 0x414 | Status indicating that *LFCLKSTART* task has been triggered |
| *LFCLKSTAT* | 0x418 | Which LFCLK source is running |
| *LFCLKSRCCOPY* | 0x41C | Copy of *LFCLKSRC* register, set when *LFCLKSTART* task was triggered |
| *LFCLKSRC* | 0x518 | Clock source for the LFCLK |
| *CTIV* | 0x538 | Calibration timer interval |
| *XTALFREQ* | 0x550 | Crystal frequency |

## 13.3 Register Details

**Table 64: INTENSET**

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | HFCLKSTARTED | | | Write '1' to Enable interrupt on *HFCLKSTARTED* event. |
| | | | Enabled | 1 | Enable |
| B | RW | LFCLKSTARTED | | | Write '1' to Enable interrupt on *LFCLKSTARTED* event. |

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Enabled | 1 | Enable |
| C | RW | DONE | | | Write '1' to Enable interrupt on *DONE* event. |
| | | | Enabled | 1 | Enable |
| D | RW | CTTO | | | Write '1' to Enable interrupt on *CTTO* event. |
| | | | Enabled | 1 | Enable |

## Table 65: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | HFCLKSTARTED | | | Write '1' to Clear interrupt on *HFCLKSTARTED* event. |
| | | | Disabled | 1 | Disable |
| B | RW | LFCLKSTARTED | | | Write '1' to Clear interrupt on *LFCLKSTARTED* event. |
| | | | Disabled | 1 | Disable |
| C | RW | DONE | | | Write '1' to Clear interrupt on *DONE* event. |
| | | | Disabled | 1 | Disable |
| D | RW | CTTO | | | Write '1' to Clear interrupt on *CTTO* event. |
| | | | Disabled | 1 | Disable |

## Table 66: HFCLKRUN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | STATUS | | | *HFCLKSTART* task triggered or not |
| | | | NotTriggered | 0 | Task not triggered |
| | | | Triggered | 1 | Task triggered |

## Table 67: HFCLKSTAT

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | SRC | | | Active clock source |
| | | | RC | 0 | 16 MHz RC oscillator running and generating the HFCLK |
| | | | Xtal | 1 | 16 MHz HFCLK crystal oscillator running and generating the HFCLK |
| B | R | STATE | | | HFCLK state |
| | | | NotRunning | 0 | HFCLK not running |
| | | | Running | 1 | HFCLK running |

## Table 68: LFCLKRUN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | STATUS | | | *LFCLKSTART* task triggered or not |
| | | | NotTriggered | 0 | Task not triggered |
| | | | Triggered | 1 | Task triggered |

## Table 69: LFCLKSTAT

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | SRC | | | Active clock source |
| | | | RC | 0 | 32.768 kHz RC oscillator running and generating the LFCLK |
| | | | Xtal | 1 | 32.768 kHz crystal oscillator running and generating the LFCLK |
| | | | Synth | 2 | 32.768 kHz synthesizer synthesizing 32.768 kHz (from HFCLK) and generating the LFCLK |
| B | R | STATE | | | LFCLK state |
| | | | NotRunning | 0 | LFCLK not running |
| | | | Running | 1 | LFCLK running |

### Table 70: LFCLKSRCCOPY

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | SRC | | | Clock source |
| | | | RC | 0 | 32.768 kHz RC oscillator |
| | | | Xtal | 1 | 32.768 kHz crystal oscillator |
| | | | Synth | 2 | 32.768 kHz synthesized from HFCLK |

### Table 71: LFCLKSRC

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | SRC | | | Clock source |
| | | | RC | 0 | 32.768 kHz RC oscillator |
| | | | Xtal | 1 | 32.768 kHz crystal oscillator or external clock source |
| | | | Synth | 2 | 32.768 kHz synthesized from HFCLK |

### Table 72: CTIV

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CTIV | | | Calibration timer interval in multiple of 0.25 seconds. Range: 0.25 seconds to 31.75 seconds. |

### Table 73: XTALFREQ

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | 1 | 1 | 1 | 1 | 1 | 1 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | XTALFREQ | | | Select nominal frequency of external crystal for HFCLK. This register has to match the actual crystal used in design to enable correct behaviour. |
| | | | 16MHz | 0xFF | 16 MHz crystal is used |
| | | | 32MHz | 0x00 | 32 MHz crystal is used |

# 14 General-Purpose Input/Output (GPIO)

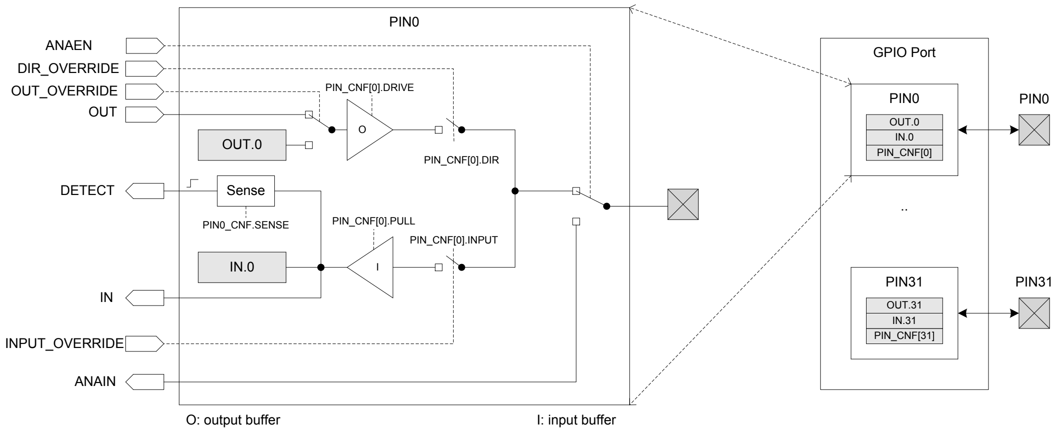



**Figure 15: GPIO Port and the GPIO pin details**

*Figure 15: GPIO Port and the GPIO pin details* on page 56 illustrates the GPIO port containing 32 individual pins, where **PIN0** is illustrated in more detail as a reference. All the signals on the left side of the illustration are used by other peripherals in the system, and therefore, are not directly available to the CPU.

## 14.1 Functional description

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* chapter for more information about retained registers.

Pins can be individually configured, through the pin sense mechanism, 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, and the default behaviour is that the DETECT signal from all pins in the GPIO Port are combined into a common DETECT signal that is routed throughout the system, which then can be utilized by other peripherals, see *Figure 15: GPIO Port and the GPIO pin details* on page 56. This mechanism is functional in both ON and OFF mode.

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

- POWER: uses the DETECT signal to exit from System OFF.
- GPIOTE: uses the DETECT signal to generate the PORT event.

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 *Figure 15: GPIO Port and the GPIO pin details* on page 56. Inputs must be connected in order 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 attach themselves to GPIO pins and override their output value and configuration, or read their analog or digital input value, see *Figure 15: GPIO Port and the GPIO pin details* on page 56.

Selected PINs also support analog input signals, see ANAIN in *Figure 15: GPIO Port and the GPIO pin details* on page 56. Pins that support analog input signals vary between devices, see the product specification for your device for more details.

## 14.2 Register Overview

### Table 74: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x50000000 | GPIO | GPIO | GPIO Port |

### Table 75: Register Overview

| Register | Offset | Description |
|---|---|---|
| Registers | | |
| *OUT* | 0x504 | Write GPIO port |
| *OUTSET* | 0x508 | Set individual bits in GPIO port |
| *OUTCLR* | 0x50C | Clear individual bits in GPIO port |
| *IN* | 0x510 | Read GPIO port |
| *DIR* | 0x514 | Direction of GPIO pins |
| *DIRSET* | 0x518 | DIR set register |
| *DIRCLR* | 0x51C | DIR clear register |
| *PIN_CNF[0]* | 0x700 | Configuration of GPIO pins |
| *PIN_CNF[1]* | 0x704 | Configuration of GPIO pins |
| *PIN_CNF[2]* | 0x708 | Configuration of GPIO pins |
| *PIN_CNF[3]* | 0x70C | Configuration of GPIO pins |
| *PIN_CNF[4]* | 0x710 | Configuration of GPIO pins |
| *PIN_CNF[5]* | 0x714 | Configuration of GPIO pins |
| *PIN_CNF[6]* | 0x718 | Configuration of GPIO pins |
| *PIN_CNF[7]* | 0x71C | Configuration of GPIO pins |
| *PIN_CNF[8]* | 0x720 | Configuration of GPIO pins |
| *PIN_CNF[9]* | 0x724 | Configuration of GPIO pins |
| *PIN_CNF[10]* | 0x728 | Configuration of GPIO pins |
| *PIN_CNF[11]* | 0x72C | Configuration of GPIO pins |
| *PIN_CNF[12]* | 0x730 | Configuration of GPIO pins |
| *PIN_CNF[13]* | 0x734 | Configuration of GPIO pins |
| *PIN_CNF[14]* | 0x738 | Configuration of GPIO pins |
| *PIN_CNF[15]* | 0x73C | Configuration of GPIO pins |
| *PIN_CNF[16]* | 0x740 | Configuration of GPIO pins |
| *PIN_CNF[17]* | 0x744 | Configuration of GPIO pins |
| *PIN_CNF[18]* | 0x748 | Configuration of GPIO pins |
| *PIN_CNF[19]* | 0x74C | Configuration of GPIO pins |
| *PIN_CNF[20]* | 0x750 | Configuration of GPIO pins |
| *PIN_CNF[21]* | 0x754 | Configuration of GPIO pins |
| *PIN_CNF[22]* | 0x758 | Configuration of GPIO pins |
| *PIN_CNF[23]* | 0x75C | Configuration of GPIO pins |
| *PIN_CNF[24]* | 0x760 | Configuration of GPIO pins |
| *PIN_CNF[25]* | 0x764 | Configuration of GPIO pins |
| *PIN_CNF[26]* | 0x768 | Configuration of GPIO pins |
| *PIN_CNF[27]* | 0x76C | Configuration of GPIO pins |
| *PIN_CNF[28]* | 0x770 | Configuration of GPIO pins |
| *PIN_CNF[29]* | 0x774 | Configuration of GPIO pins |
| *PIN_CNF[30]* | 0x778 | Configuration of GPIO pins |
| *PIN_CNF[31]* | 0x77C | Configuration of GPIO pins |

## 14.3 Register Details

### Table 76: OUT

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PIN0 | | | Pin 0 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| B | RW | PIN1 | | | Pin 1 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| C | RW | PIN2 | | | Pin 2 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| D | RW | PIN3 | | | Pin 3 |
| | | | Low | 0 | Pin driver 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 | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | High | 1 | Pin driver is high |
| E | RW | PIN4 | | | Pin 4 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| F | RW | PIN5 | | | Pin 5 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| G | RW | PIN6 | | | Pin 6 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| H | RW | PIN7 | | | Pin 7 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| I | RW | PIN8 | | | Pin 8 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| J | RW | PIN9 | | | Pin 9 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| K | RW | PIN10 | | | Pin 10 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| L | RW | PIN11 | | | Pin 11 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| M | RW | PIN12 | | | Pin 12 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| N | RW | PIN13 | | | Pin 13 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| O | RW | PIN14 | | | Pin 14 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| P | RW | PIN15 | | | Pin 15 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| Q | RW | PIN16 | | | Pin 16 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| R | RW | PIN17 | | | Pin 17 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| S | RW | PIN18 | | | Pin 18 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| T | RW | PIN19 | | | Pin 19 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| U | RW | PIN20 | | | Pin 20 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| V | RW | PIN21 | | | Pin 21 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| W | RW | PIN22 | | | Pin 22 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| X | RW | PIN23 | | | Pin 23 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| Y | RW | PIN24 | | | Pin 24 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| Z | RW | PIN25 | | | Pin 25 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| AA | RW | PIN26 | | | Pin 26 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| AB | RW | PIN27 | | | Pin 27 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| AC | RW | PIN28 | | | Pin 28 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| AD | RW | PIN29 | | | Pin 29 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| AE | RW | PIN30 | | | Pin 30 |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |
| AF | RW | PIN31 | | | Pin 31 |
| | | | Low | 0 | Pin driver is low |
| | | | High | 1 | Pin driver is high |

## Table 77: OUTSET

**Note:** Read: reads value of OUT register.
**Note:** Individual bits are set by writing a '1' to the bits that shall be set. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PIN0 | | | Pin 0 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| B | RW | PIN1 | | | Pin 1 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| C | RW | PIN2 | | | Pin 2 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| D | RW | PIN3 | | | Pin 3 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| E | RW | PIN4 | | | Pin 4 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| F | RW | PIN5 | | | Pin 5 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| G | RW | PIN6 | | | Pin 6 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| H | RW | PIN7 | | | Pin 7 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| I | RW | PIN8 | | | Pin 8 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| J | RW | PIN9 | | | Pin 9 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| K | RW | PIN10 | | | Pin 10 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| L | RW | PIN11 | | | Pin 11 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| M | RW | PIN12 | | | Pin 12 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| N | RW | PIN13 | | | Pin 13 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| O | RW | PIN14 | | | Pin 14 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| P | RW | PIN15 | | | Pin 15 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| Q | RW | PIN16 | | | Pin 16 |

**Note:** Read: reads value of OUT register.
**Note:** Individual bits are set by writing a '1' to the bits that shall be set. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| R | RW | PIN17 | | | Pin 17 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| S | RW | PIN18 | | | Pin 18 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| T | RW | PIN19 | | | Pin 19 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| U | RW | PIN20 | | | Pin 20 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| V | RW | PIN21 | | | Pin 21 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| W | RW | PIN22 | | | Pin 22 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| X | RW | PIN23 | | | Pin 23 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| Y | RW | PIN24 | | | Pin 24 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| Z | RW | PIN25 | | | Pin 25 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| AA | RW | PIN26 | | | Pin 26 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| AB | RW | PIN27 | | | Pin 27 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| AC | RW | PIN28 | | | Pin 28 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| AD | RW | PIN29 | | | Pin 29 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| AE | RW | PIN30 | | | Pin 30 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |
| AF | RW | PIN31 | | | Pin 31 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Set | 1 | Write: writing a '1' sets the pin high |

## Table 78: OUTCLR

**Note:** Read: reads value of OUT register.
**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PIN0 | | | Pin 0 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| B | RW | PIN1 | | | Pin 1 |

**Note:** Read: reads value of OUT register.

**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | | | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | | | | AF | AE | AD | AC | AB | AA | 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 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| C | RW | PIN2 | | | Pin 2 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| D | RW | PIN3 | | | Pin 3 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| E | RW | PIN4 | | | Pin 4 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| F | RW | PIN5 | | | Pin 5 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| G | RW | PIN6 | | | Pin 6 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| H | RW | PIN7 | | | Pin 7 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| I | RW | PIN8 | | | Pin 8 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| J | RW | PIN9 | | | Pin 9 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| K | RW | PIN10 | | | Pin 10 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| L | RW | PIN11 | | | Pin 11 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| M | RW | PIN12 | | | Pin 12 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| N | RW | PIN13 | | | Pin 13 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| O | RW | PIN14 | | | Pin 14 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| P | RW | PIN15 | | | Pin 15 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| Q | RW | PIN16 | | | Pin 16 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| R | RW | PIN17 | | | Pin 17 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| S | RW | PIN18 | | | Pin 18 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| T | RW | PIN19 | | | Pin 19 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| U | RW | PIN20 | | | Pin 20 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |

**Note:** Read: reads value of OUT register.
**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| V | RW | PIN21 | | | Pin 21 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| W | RW | PIN22 | | | Pin 22 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| X | RW | PIN23 | | | Pin 23 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| Y | RW | PIN24 | | | Pin 24 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| Z | RW | PIN25 | | | Pin 25 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| AA | RW | PIN26 | | | Pin 26 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| AB | RW | PIN27 | | | Pin 27 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| AC | RW | PIN28 | | | Pin 28 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| AD | RW | PIN29 | | | Pin 29 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| AE | RW | PIN30 | | | Pin 30 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |
| AF | RW | PIN31 | | | Pin 31 |
| | | | Low | 0 | Read: pin driver is low |
| | | | High | 1 | Read: pin driver is high |
| | | | Clear | 1 | Write: writing a '1' sets the pin low |

## Table 79: IN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | PIN0 | | | Pin 0 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| B | R | PIN1 | | | Pin 1 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| C | R | PIN2 | | | Pin 2 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| D | R | PIN3 | | | Pin 3 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| E | R | PIN4 | | | Pin 4 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| F | R | PIN5 | | | Pin 5 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| G | R | PIN6 | | | Pin 6 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| H | R | PIN7 | | | Pin 7 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| I | R | PIN8 | | | Pin 8 |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| J | R | PIN9 | | | Pin 9 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| K | R | PIN10 | | | Pin 10 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| L | R | PIN11 | | | Pin 11 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| M | R | PIN12 | | | Pin 12 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| N | R | PIN13 | | | Pin 13 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| O | R | PIN14 | | | Pin 14 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| P | R | PIN15 | | | Pin 15 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| Q | R | PIN16 | | | Pin 16 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| R | R | PIN17 | | | Pin 17 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| S | R | PIN18 | | | Pin 18 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| T | R | PIN19 | | | Pin 19 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| U | R | PIN20 | | | Pin 20 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| V | R | PIN21 | | | Pin 21 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| W | R | PIN22 | | | Pin 22 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| X | R | PIN23 | | | Pin 23 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| Y | R | PIN24 | | | Pin 24 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| Z | R | PIN25 | | | Pin 25 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| AA | R | PIN26 | | | Pin 26 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| AB | R | PIN27 | | | Pin 27 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| AC | R | PIN28 | | | Pin 28 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| AD | R | PIN29 | | | Pin 29 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| AE | R | PIN30 | | | Pin 30 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |
| AF | R | PIN31 | | | Pin 31 |
| | | | Low | 0 | Pin input is low |
| | | | High | 1 | Pin input is high |

## Table 80: DIR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PIN0 | | | Pin 0 |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| B | RW | PIN1 | | | Pin 1 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| C | RW | PIN2 | | | Pin 2 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| D | RW | PIN3 | | | Pin 3 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| E | RW | PIN4 | | | Pin 4 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| F | RW | PIN5 | | | Pin 5 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| G | RW | PIN6 | | | Pin 6 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| H | RW | PIN7 | | | Pin 7 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| I | RW | PIN8 | | | Pin 8 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| J | RW | PIN9 | | | Pin 9 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| K | RW | PIN10 | | | Pin 10 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| L | RW | PIN11 | | | Pin 11 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| M | RW | PIN12 | | | Pin 12 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| N | RW | PIN13 | | | Pin 13 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| O | RW | PIN14 | | | Pin 14 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| P | RW | PIN15 | | | Pin 15 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| Q | RW | PIN16 | | | Pin 16 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| R | RW | PIN17 | | | Pin 17 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| S | RW | PIN18 | | | Pin 18 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| T | RW | PIN19 | | | Pin 19 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| U | RW | PIN20 | | | Pin 20 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| V | RW | PIN21 | | | Pin 21 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| W | RW | PIN22 | | | Pin 22 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| X | RW | PIN23 | | | Pin 23 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| Y | RW | PIN24 | | | Pin 24 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| Z | RW | PIN25 | | | Pin 25 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| AA | RW | PIN26 | | | Pin 26 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| AB | RW | PIN27 | | | Pin 27 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| AC | RW | PIN28 | | | Pin 28 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| AD | RW | PIN29 | | | Pin 29 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| AE | RW | PIN30 | | | Pin 30 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |
| AF | RW | PIN31 | | | Pin 31 |
| | | | Input | 0 | Pin set as input |
| | | | Output | 1 | Pin set as output |

## Table 81: DIRSET

**Note:** Read: reads value of DIR register.
**Note:** Individual bits are set by writing a '1' to the bits that shall be set. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PIN0 | | | Set as output pin 0 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| B | RW | PIN1 | | | Set as output pin 1 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| C | RW | PIN2 | | | Set as output pin 2 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| D | RW | PIN3 | | | Set as output pin 3 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| E | RW | PIN4 | | | Set as output pin 4 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| F | RW | PIN5 | | | Set as output pin 5 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| G | RW | PIN6 | | | Set as output pin 6 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| H | RW | PIN7 | | | Set as output pin 7 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| I | RW | PIN8 | | | Set as output pin 8 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| J | RW | PIN9 | | | Set as output pin 9 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| K | RW | PIN10 | | | Set as output pin 10 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| L | RW | PIN11 | | | Set as output pin 11 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| M | RW | PIN12 | | | Set as output pin 12 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| N | RW | PIN13 | | | Set as output pin 13 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |

**Note:** Read: reads value of DIR register.
**Note:** Individual bits are set by writing a '1' to the bits that shall be set. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| O | RW | PIN14 | | | Set as output pin 14 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| P | RW | PIN15 | | | Set as output pin 15 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| Q | RW | PIN16 | | | Set as output pin 16 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| R | RW | PIN17 | | | Set as output pin 17 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| S | RW | PIN18 | | | Set as output pin 18 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| T | RW | PIN19 | | | Set as output pin 19 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| U | RW | PIN20 | | | Set as output pin 20 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| V | RW | PIN21 | | | Set as output pin 21 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| W | RW | PIN22 | | | Set as output pin 22 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| X | RW | PIN23 | | | Set as output pin 23 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| Y | RW | PIN24 | | | Set as output pin 24 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| Z | RW | PIN25 | | | Set as output pin 25 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| AA | RW | PIN26 | | | Set as output pin 26 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| AB | RW | PIN27 | | | Set as output pin 27 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| AC | RW | PIN28 | | | Set as output pin 28 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| AD | RW | PIN29 | | | Set as output pin 29 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| AE | RW | PIN30 | | | Set as output pin 30 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |
| AF | RW | PIN31 | | | Set as output pin 31 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Set | 1 | Write: writing a '1' sets pin to output |

### Table 82: DIRCLR

**Note:** Read: reads value of DIR register.
**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PIN0 | | | Set as input pin 0 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| B | RW | PIN1 | | | Set as input pin 1 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| C | RW | PIN2 | | | Set as input pin 2 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| D | RW | PIN3 | | | Set as input pin 3 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| E | RW | PIN4 | | | Set as input pin 4 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| F | RW | PIN5 | | | Set as input pin 5 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| G | RW | PIN6 | | | Set as input pin 6 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| H | RW | PIN7 | | | Set as input pin 7 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| I | RW | PIN8 | | | Set as input pin 8 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| J | RW | PIN9 | | | Set as input pin 9 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| K | RW | PIN10 | | | Set as input pin 10 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| L | RW | PIN11 | | | Set as input pin 11 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| M | RW | PIN12 | | | Set as input pin 12 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| N | RW | PIN13 | | | Set as input pin 13 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| O | RW | PIN14 | | | Set as input pin 14 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| P | RW | PIN15 | | | Set as input pin 15 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| Q | RW | PIN16 | | | Set as input pin 16 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| R | RW | PIN17 | | | Set as input pin 17 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| S | RW | PIN18 | | | Set as input pin 18 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |

**Note:** Read: reads value of DIR register.

**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| T | RW | PIN19 | | | Set as input pin 19 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| U | RW | PIN20 | | | Set as input pin 20 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| V | RW | PIN21 | | | Set as input pin 21 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| W | RW | PIN22 | | | Set as input pin 22 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| X | RW | PIN23 | | | Set as input pin 23 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| Y | RW | PIN24 | | | Set as input pin 24 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| Z | RW | PIN25 | | | Set as input pin 25 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| AA | RW | PIN26 | | | Set as input pin 26 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| AB | RW | PIN27 | | | Set as input pin 27 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| AC | RW | PIN28 | | | Set as input pin 28 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| AD | RW | PIN29 | | | Set as input pin 29 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| AE | RW | PIN30 | | | Set as input pin 30 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |
| AF | RW | PIN31 | | | Set as input pin 31 |
| | | | Input | 0 | Read: pin set as input |
| | | | Output | 1 | Read: pin set as output |
| | | | Clear | 1 | Write: writing a '1' sets pin to input |

## Table 83: PIN_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 | | | | | | | | | | | | | | | E | E | | | | | | D | D | D | | | | | C | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | DIR | | | Pin direction |
| | | | 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' |
| | | | D0H1 | 5 | Disconnect '0', high drive '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 | | | | | | | | | | | | | | | E | E | | | | | | D | D | D | | | | | C | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| | | | S0D1 | 6 | Standard '0'. disconnect '1' |
| | | | H0D1 | 7 | High drive '0', disconnect '1' |
| E | RW | SENSE | | | Pin sensing mechanism |
| | | | Disabled | 0 | Disabled |
| | | | High | 2 | Sense for high level |
| | | | Low | 3 | Sense for low level |

# 15 GPIO tasks and events (GPIOTE)

## 15.1 Functional description

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 task can be used in each GPIOTE channel for performing the following write operations to a pin:

- 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

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

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

When an OUT[n] task or an IN[n] event has been configured to operate on a pin, the pin can only be written from the GPIOTE module. Attempting to write a pin as a normal GPIO pin will have no effect.

As long as an 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. 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 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].

### 15.1.2 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* chapter for more information about the DETECT signal.

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.

### 15.1.3 Task 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 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 setting in GPIO. When Disabled is selected in CONFIG.MODE, the pin specified by CONFIG.PSEL will use its configuration from the PIN_CNF registers in GPIO.

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

## 15.2 Register Overview

### Table 84: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40006000 | GPIOTE | GPIOTE | GPIO Tasks and Events |

### Table 85: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *OUT[0]* | 0x000 | Task for writing to pin specified in CONFIG[0].PSEL. Action on pin is configured in CONFIG[0].POLARITY. |
| *OUT[1]* | 0x004 | Task for writing to pin specified in CONFIG[1].PSEL. Action on pin is configured in CONFIG[1].POLARITY. |
| *OUT[2]* | 0x008 | Task for writing to pin specified in CONFIG[2].PSEL. Action on pin is configured in CONFIG[2].POLARITY. |
| *OUT[3]* | 0x00C | Task for writing to pin specified in CONFIG[3].PSEL. Action on pin is configured in CONFIG[3].POLARITY. |
| Events | | |
| *IN[0]* | 0x100 | Event generated from pin specified in CONFIG[0].PSEL |
| *IN[1]* | 0x104 | Event generated from pin specified in CONFIG[1].PSEL |
| *IN[2]* | 0x108 | Event generated from pin specified in CONFIG[2].PSEL |
| *IN[3]* | 0x10C | Event generated from pin specified in CONFIG[3].PSEL |
| *PORT* | 0x17C | Event generated from multiple input pins |
| Registers | | |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *CONFIG[0]* | 0x510 | Configuration for OUT[n] task and IN[n] event |
| *CONFIG[1]* | 0x514 | Configuration for OUT[n] task and IN[n] event |
| *CONFIG[2]* | 0x518 | Configuration for OUT[n] task and IN[n] event |
| *CONFIG[3]* | 0x51C | Configuration for OUT[n] task and IN[n] event |

## 15.3 Register Details

### Table 86: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | IN0 | | | Enable or disable interrupt on *IN[0]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | IN1 | | | Enable or disable interrupt on *IN[1]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| C | RW | IN2 | | | Enable or disable interrupt on *IN[2]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| D | RW | IN3 | | | Enable or disable interrupt on *IN[3]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| I | RW | PORT | | | Enable or disable interrupt on *PORT* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 87: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | IN0 | | | Write '1' to Enable interrupt on *IN[0]* event. |
| | | | Enabled | 1 | Enable |
| B | RW | IN1 | | | Write '1' to Enable interrupt on *IN[1]* event. |
| | | | Enabled | 1 | Enable |
| C | RW | IN2 | | | Write '1' to Enable interrupt on *IN[2]* event. |
| | | | Enabled | 1 | Enable |
| D | RW | IN3 | | | Write '1' to Enable interrupt on *IN[3]* event. |
| | | | Enabled | 1 | Enable |
| I | RW | PORT | | | Write '1' to Enable interrupt on *PORT* event. |
| | | | Enabled | 1 | Enable |

### Table 88: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | IN0 | | | Write '1' to Clear interrupt on *IN[0]* event. |
| | | | Disabled | 1 | Disable |
| B | RW | IN1 | | | Write '1' to Clear interrupt on *IN[1]* event. |
| | | | Disabled | 1 | Disable |
| C | RW | IN2 | | | Write '1' to Clear interrupt on *IN[2]* event. |
| | | | Disabled | 1 | Disable |
| D | RW | IN3 | | | Write '1' to Clear interrupt on *IN[3]* event. |
| | | | Disabled | 1 | Disable |
| I | RW | PORT | | | Write '1' to Clear interrupt on *PORT* event. |
| | | | Disabled | 1 | Disable |

### Table 89: CONFIG[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 | | | | | | | | | | | | D | | | C | C | | | | B | B | B | B | B | | | | | | | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | MODE | | | Mode |
| | | | Disabled | 0 | Disabled. Pin specified by PSEL will not be acquired by the GPIOTE module. |
| | | | Event | 1 | Event mode<br>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<br>The pin specified by PSEL will be configured as an output and triggering the 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] | Pin number associated with OUT[n] task and IN[n] event |
| C | 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. |
| D | 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 |
| | | | High | 1 | Task mode: Initial value of pin before task triggering is high |

# 16 Programmable Peripheral Interconnect (PPI)

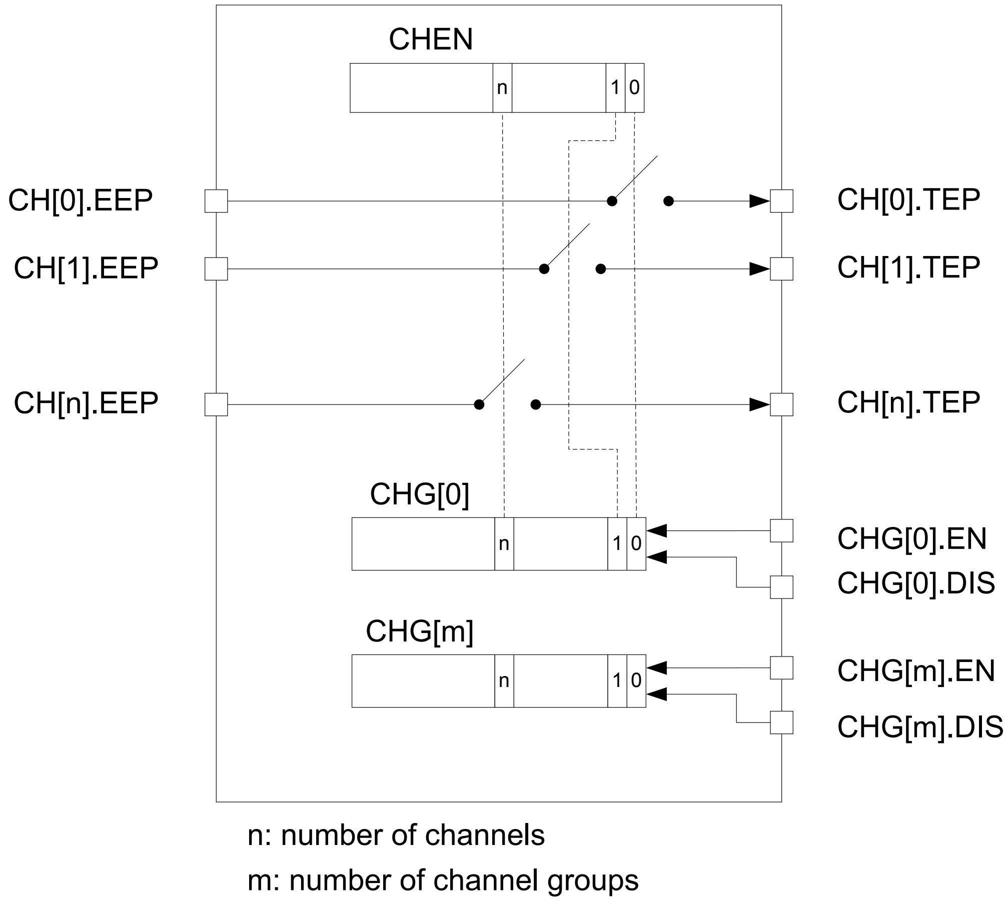



**Figure 16: PPI block diagram**

## 16.1 Functional description

The Programmable Peripheral Interconnect (PPI) enables different peripherals to interact autonomously with each other using tasks and events and without having to use the CPU.

The PPI provides a mechanism to automatically trigger a task in one peripheral as a result of an event occurring in another peripheral. A task is connected to an event through a PPI channel. The PPI channel is composed of two end-point registers, the Event End-Point (EEP) and the Task End-Point (TEP). A peripheral task is connected to a Task End-Point using the address of the task register associated with the task. Similarly, a peripheral event is connected to an Event End-Point using the address of the event register associated with the event.

There are two ways of enabling and disabling PPI channels:

- Enable or disable PPI channels individually using the CHEN, CHENSET, and CHENCLR registers.
- Enable or disable PPI channels in PPI channel groups through the groups' ENABLE and DISABLE tasks. Prior to these tasks being triggered, the PPI channel group must be configured to define which PPI channels belongs to which groups.

PPI tasks (for example, CHG0EN) can be triggered through the PPI like any other task, which means they can be hooked up to a PPI channel as a TEP. One event can trigger multiple tasks by using multiple channels and one task can be triggered by multiple events in the same way.

### 16.1.1 Pre-programmed channels

As illustrated in *Table 90: Pre-programmed channels* on page 74 some of the PPI's channels are pre-programmed. These channels cannot be configured by the CPU, but can be added to groups and enabled/disabled like the general purpose PPI channels.

#### Table 90: Pre-programmed channels

| Channel | EEP | TEP |
|---|---|---|
| 20 | TIMER0->EVENTS_COMPARE[0] | RADIO->TASKS_TXEN |
| 21 | TIMER0->EVENTS_COMPARE[0] | RADIO->TASKS_RXEN |
| 22 | TIMER0->EVENTS_COMPARE[1] | RADIO->TASKS_DISABLE |
| 23 | RADIO->EVENTS_BCMATCH | AAR->TASKS_START |
| 24 | RADIO->EVENTS_READY | CCM->TASKS_KSGEN |
| 25 | RADIO->EVENTS_ADDRESS | CCM->TASKS_CRYPT |
| 26 | RADIO->EVENTS_ADDRESS | TIMER0->TASKS_CAPTURE[1] |
| 27 | RADIO->EVENTS_END | TIMER0->TASKS_CAPTURE[2] |
| 28 | RTC0->EVENTS_COMPARE[0] | RADIO->TASKS_TXEN |
| 29 | RTC0->EVENTS_COMPARE[0] | RADIO->TASKS_RXEN |
| 30 | RTC0->EVENTS_COMPARE[0] | TIMER0->TASKS_CLEAR |
| 31 | RTC0->EVENTS_COMPARE[0] | TIMER0->TASKS_START |

## 16.2 Register Overview

#### Table 91: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x4001F000 | PPI | PPI | Programmable Peripheral Interconnect |

#### Table 92: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *CHG[0].EN* | 0x000 | Enable channel group 0 |
| *CHG[0].DIS* | 0x004 | Disable channel group 0 |
| *CHG[1].EN* | 0x008 | Enable channel group 1 |
| *CHG[1].DIS* | 0x00C | Disable channel group 1 |
| *CHG[2].EN* | 0x010 | Enable channel group 2 |
| *CHG[2].DIS* | 0x014 | Disable channel group 2 |
| *CHG[3].EN* | 0x018 | Enable channel group 3 |
| *CHG[3].DIS* | 0x01C | Disable channel group 3 |
| Registers | | |
| *CHEN* | 0x500 | Channel enable register |
| *CHENSET* | 0x504 | Channel enable set register |
| *CHENCLR* | 0x508 | Channel enable clear register |
| *CH[0].EEP* | 0x510 | Channel 0 event end-point |
| *CH[0].TEP* | 0x514 | Channel 0 task end-point |
| *CH[1].EEP* | 0x518 | Channel 1 event end-point |
| *CH[1].TEP* | 0x51C | Channel 1 task end-point |
| *CH[2].EEP* | 0x520 | Channel 2 event end-point |
| *CH[2].TEP* | 0x524 | Channel 2 task end-point |
| *CH[3].EEP* | 0x528 | Channel 3 event end-point |
| *CH[3].TEP* | 0x52C | Channel 3 task end-point |
| *CH[4].EEP* | 0x530 | Channel 4 event end-point |
| *CH[4].TEP* | 0x534 | Channel 4 task end-point |
| *CH[5].EEP* | 0x538 | Channel 5 event end-point |
| *CH[5].TEP* | 0x53C | Channel 5 task end-point |
| *CH[6].EEP* | 0x540 | Channel 6 event end-point |
| *CH[6].TEP* | 0x544 | Channel 6 task end-point |
| *CH[7].EEP* | 0x548 | Channel 7 event end-point |
| *CH[7].TEP* | 0x54C | Channel 7 task end-point |
| *CH[8].EEP* | 0x550 | Channel 8 event end-point |
| *CH[8].TEP* | 0x554 | Channel 8 task end-point |
| *CH[9].EEP* | 0x558 | Channel 9 event end-point |
| *CH[9].TEP* | 0x55C | Channel 9 task end-point |
| *CH[10].EEP* | 0x560 | Channel 10 event end-point |
| *CH[10].TEP* | 0x564 | Channel 10 task end-point |
| *CH[11].EEP* | 0x568 | Channel 11 event end-point |
| *CH[11].TEP* | 0x56C | Channel 11 task end-point |
| *CH[12].EEP* | 0x570 | Channel 12 event end-point |
| *CH[12].TEP* | 0x574 | Channel 12 task end-point |
| *CH[13].EEP* | 0x578 | Channel 13 event end-point |
| *CH[13].TEP* | 0x57C | Channel 13 task end-point |
| *CH[14].EEP* | 0x580 | Channel 14 event end-point |
| *CH[14].TEP* | 0x584 | Channel 14 task end-point |
| *CH[15].EEP* | 0x588 | Channel 15 event end-point |
| *CH[15].TEP* | 0x58C | Channel 15 task end-point |

| Register | Offset | Description |
|---|---|---|
| *CHG[0]* | 0x800 | Channel group 0 |
| *CHG[1]* | 0x804 | Channel group 1 |
| *CHG[2]* | 0x808 | Channel group 2 |
| *CHG[3]* | 0x80C | Channel group 3 |

# 16.3 Register Details

## Table 93: CHEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CH0 | | | Enable or disable channel 0 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| B | RW | CH1 | | | Enable or disable channel 1 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| C | RW | CH2 | | | Enable or disable channel 2 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| D | RW | CH3 | | | Enable or disable channel 3 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| E | RW | CH4 | | | Enable or disable channel 4 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| F | RW | CH5 | | | Enable or disable channel 5 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| G | RW | CH6 | | | Enable or disable channel 6 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| H | RW | CH7 | | | Enable or disable channel 7 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| I | RW | CH8 | | | Enable or disable channel 8 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| J | RW | CH9 | | | Enable or disable channel 9 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| K | RW | CH10 | | | Enable or disable channel 10 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| L | RW | CH11 | | | Enable or disable channel 11 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| M | RW | CH12 | | | Enable or disable channel 12 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| N | RW | CH13 | | | Enable or disable channel 13 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| O | RW | CH14 | | | Enable or disable channel 14 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| P | RW | CH15 | | | Enable or disable channel 15 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| U | RW | CH20 | | | Enable or disable channel 20 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| V | RW | CH21 | | | Enable or disable channel 21 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| W | RW | CH22 | | | Enable or disable channel 22 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| X | RW | CH23 | | | Enable or disable channel 23 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| Y | RW | CH24 | | | Enable or disable channel 24 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| Z | RW | CH25 | | | Enable or disable channel 25 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| AA | RW | CH26 | | | Enable or disable channel 26 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| AB | RW | CH27 | | | Enable or disable channel 27 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| AC | RW | CH28 | | | Enable or disable channel 28 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| AD | RW | CH29 | | | Enable or disable channel 29 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| AE | RW | CH30 | | | Enable or disable channel 30 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |
| AF | RW | CH31 | | | Enable or disable channel 31 |
| | | | Disabled | 0 | Disable channel |
| | | | Enabled | 1 | Enable channel |

## Table 94: CHENSET

**Note:** Read: reads value of CH{i} field in CHEN register.
**Note:** Individual bits are set by writing a '1' to the bits that shall be set. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CH0 | | | Write '1': Enable channel 0. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| B | RW | CH1 | | | Write '1': Enable channel 1. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| C | RW | CH2 | | | Write '1': Enable channel 2. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| D | RW | CH3 | | | Write '1': Enable channel 3. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| E | RW | CH4 | | | Write '1': Enable channel 4. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| F | RW | CH5 | | | Write '1': Enable channel 5. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| G | RW | CH6 | | | Write '1': Enable channel 6. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| H | RW | CH7 | | | Write '1': Enable channel 7. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| I | RW | CH8 | | | Write '1': Enable channel 8. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| J | RW | CH9 | | | Write '1': Enable channel 9. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| K | RW | CH10 | | | Write '1': Enable channel 10. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| L | RW | CH11 | | | Write '1': Enable channel 11. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| M | RW | CH12 | | | Write '1': Enable channel 12. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |

**Note:** Read: reads value of CH{i} field in CHEN register.
**Note:** Individual bits are set by writing a '1' to the bits that shall be set. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| N | RW | CH13 | | | Write '1': Enable channel 13. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| O | RW | CH14 | | | Write '1': Enable channel 14. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| P | RW | CH15 | | | Write '1': Enable channel 15. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| U | RW | CH20 | | | Write '1': Enable channel 20. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| V | RW | CH21 | | | Write '1': Enable channel 21. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| W | RW | CH22 | | | Write '1': Enable channel 22. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| X | RW | CH23 | | | Write '1': Enable channel 23. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| Y | RW | CH24 | | | Write '1': Enable channel 24. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| Z | RW | CH25 | | | Write '1': Enable channel 25. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| AA | RW | CH26 | | | Write '1': Enable channel 26. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| AB | RW | CH27 | | | Write '1': Enable channel 27. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| AC | RW | CH28 | | | Write '1': Enable channel 28. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| AD | RW | CH29 | | | Write '1': Enable channel 29. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| AE | RW | CH30 | | | Write '1': Enable channel 30. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |
| AF | RW | CH31 | | | Write '1': Enable channel 31. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Set | 1 | Write: Enable channel |

## Table 95: CHENCLR

**Note:** Read: reads value of CH{i} field in CHEN register.
**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CH0 | | | Write '1': Disable channel 0. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| B | RW | CH1 | | | Write '1': Disable channel 1. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |

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

**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| C | RW | CH2 | | | Write '1': Disable channel 2. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| D | RW | CH3 | | | Write '1': Disable channel 3. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| E | RW | CH4 | | | Write '1': Disable channel 4. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| F | RW | CH5 | | | Write '1': Disable channel 5. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| G | RW | CH6 | | | Write '1': Disable channel 6. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| H | RW | CH7 | | | Write '1': Disable channel 7. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| I | RW | CH8 | | | Write '1': Disable channel 8. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| J | RW | CH9 | | | Write '1': Disable channel 9. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| K | RW | CH10 | | | Write '1': Disable channel 10. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| L | RW | CH11 | | | Write '1': Disable channel 11. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| M | RW | CH12 | | | Write '1': Disable channel 12. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| N | RW | CH13 | | | Write '1': Disable channel 13. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| O | RW | CH14 | | | Write '1': Disable channel 14. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| P | RW | CH15 | | | Write '1': Disable channel 15. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| U | RW | CH20 | | | Write '1': Disable channel 20. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| V | RW | CH21 | | | Write '1': Disable channel 21. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| W | RW | CH22 | | | Write '1': Disable channel 22. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| X | RW | CH23 | | | Write '1': Disable channel 23. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| Y | RW | CH24 | | | Write '1': Disable channel 24. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| Z | RW | CH25 | | | Write '1': Disable channel 25. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |

**Note:** Read: reads value of CH{i} field in CHEN register.
**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| AA | RW | CH26 | | | Write '1': Disable channel 26. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| AB | RW | CH27 | | | Write '1': Disable channel 27. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| AC | RW | CH28 | | | Write '1': Disable channel 28. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| AD | RW | CH29 | | | Write '1': Disable channel 29. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| AE | RW | CH30 | | | Write '1': Disable channel 30. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |
| AF | RW | CH31 | | | Write '1': Disable channel 31. Write '0': no effect |
| | | | Disabled | 0 | Read: channel disabled |
| | | | Enabled | 1 | Read: channel enabled |
| | | | Clear | 1 | Write: disable channel |

## Table 96: CH[m].EEP

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | EEP | | | Pointer to event register. Accepts only addresses to registers from the Event group. |

## Table 97: CH[m].TEP

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TEP | | | Pointer to task register. Accepts only addresses to registers from the Task group. |

## Table 98: CHG[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 | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CH0 | | | Include or exclude channel 0 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| B | RW | CH1 | | | Include or exclude channel 1 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| C | RW | CH2 | | | Include or exclude channel 2 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| D | RW | CH3 | | | Include or exclude channel 3 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| E | RW | CH4 | | | Include or exclude channel 4 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| F | RW | CH5 | | | Include or exclude channel 5 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| G | RW | CH6 | | | Include or exclude channel 6 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| H | RW | CH7 | | | Include or exclude channel 7 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | AF | AE | AD | AC | AB | AA | Z | Y | X | W | V | U | | | | | P | O | N | M | L | K | J | I | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| I | RW | CH8 | | | Include or exclude channel 8 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| J | RW | CH9 | | | Include or exclude channel 9 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| K | RW | CH10 | | | Include or exclude channel 10 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| L | RW | CH11 | | | Include or exclude channel 11 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| M | RW | CH12 | | | Include or exclude channel 12 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| N | RW | CH13 | | | Include or exclude channel 13 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| O | RW | CH14 | | | Include or exclude channel 14 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| P | RW | CH15 | | | Include or exclude channel 15 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| U | RW | CH20 | | | Include or exclude channel 20 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| V | RW | CH21 | | | Include or exclude channel 21 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| W | RW | CH22 | | | Include or exclude channel 22 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| X | RW | CH23 | | | Include or exclude channel 23 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| Y | RW | CH24 | | | Include or exclude channel 24 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| Z | RW | CH25 | | | Include or exclude channel 25 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| AA | RW | CH26 | | | Include or exclude channel 26 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| AB | RW | CH27 | | | Include or exclude channel 27 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| AC | RW | CH28 | | | Include or exclude channel 28 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| AD | RW | CH29 | | | Include or exclude channel 29 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| AE | RW | CH30 | | | Include or exclude channel 30 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |
| AF | RW | CH31 | | | Include or exclude channel 31 |
| | | | Excluded | 0 | Exclude |
| | | | Included | 1 | Include |

# 17 2.4 GHz Radio (RADIO)

The RADIO contains a 2.4 GHz radio receiver and a 2.4 GHz radio transmitter that is compatible with Nordic's proprietary 250 kbps, 1 Mbps and 2 Mbps radio modes in addition to 1 Mbps Bluetooth Low Energy mode.

The RADIO implements EasyDMA. EasyDMA in combination with an automated packet assembler and packet disassembler, and an automated CRC generator and CRC checker, makes it very easy to configure and use the RADIO. See *Figure 17: RADIO block diagram* on page 81 for more information.

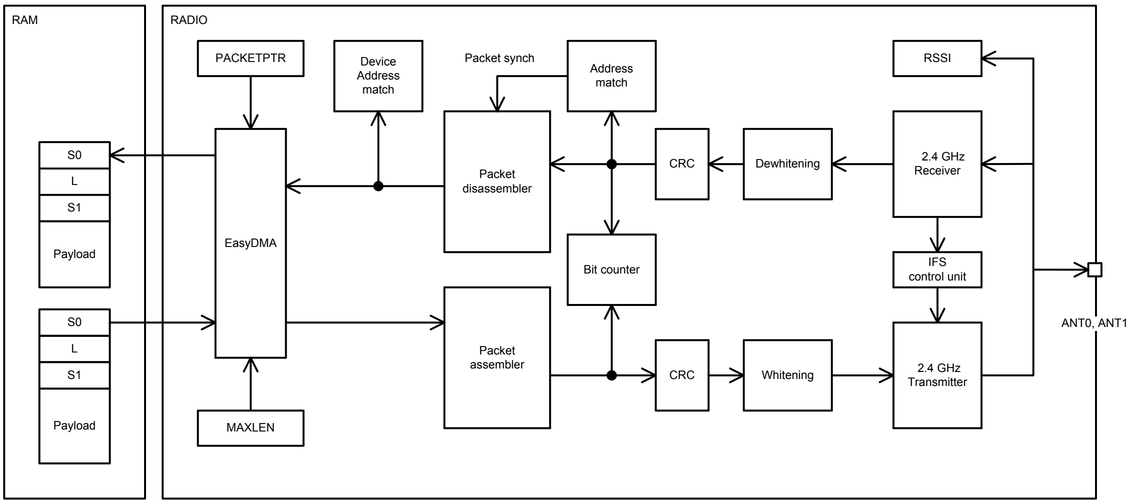



**Figure 17: RADIO block diagram**

The RADIO includes a Device Address Match unit and an interframe spacing control unit that can be utilized to simplify address white listing and interframe spacing respectively, in *Bluetooth* low energy and similar applications.

The RADIO also includes a Received Signal Strength Indicator (RSSI) and a bit counter. The bit counter generates events when a preconfigured number of bits have been sent or received by the RADIO.

## 17.1 Functional description

### 17.1.1 EasyDMA

The RADIO implements EasyDMA for reading and writing of data packets from and to the RAM without CPU involvement.

As illustrated in *Figure 17: RADIO block diagram* on page 81, the RADIO's EasyDMA utilizes the same PACKETPTR pointer for receiving packets and transmitting packets. The CPU should reconfigure this pointer every time before the RADIO is started via the START task.

The MAXLEN field in the PCNF1 register configures the maximum packet payload size in number of bytes that can be transmitted or received by the RADIO. This feature can be used to ensure that the RADIO does not overwrite, or read beyond, the RAM assigned to the packet payload. This means that if the packet payload length defined by PCNF1.STATLEN and the LENGTH field in the packet specifies a packet larger than MAXLEN, the payload will be truncated at MAXLEN.

If the payload length is specified larger than MAXLEN, the RADIO will still transmit or receive in the same way as before except the payload is now truncated to MAXLEN. The packet's LENGTH field will not be altered when the payload is truncated. The RADIO will calculate CRC as if the packet length is equal to MAXLEN.

If the PACKETPTR is not pointing to the Data RAM region, an EasyDMA transfer may result in a HardFault or RAM corruption. See *Memory* on page 15 for more information about the different memory regions.

The EasyDMA will have finished accessing the RAM when the DISABLED event is generated.

### 17.1.2 Packet configuration

A Radio packet contains the following fields: PREAMBLE, ADDRESS, LENGTH, S0, S1, PAYLOAD and CRC as illustrated in *Figure 18: On-air packet layout* on page 82. The Radio sends the different fields in the packet in the order they are illustrated below, from left to right. The preamble will be sent least significant bit first on-air.

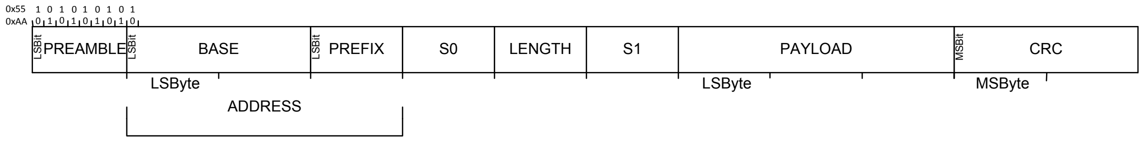



**Figure 18: On-air packet layout**

For all modes that can be specified in the MODE register, the PREAMBLE is always one byte long. If the first bit of the ADDRESS is 0 the preamble will be set to 0xAA otherwise the PREAMBLE will be set to 0x55.

Radio packets are stored in memory inside instances of a radio packet data structure as illustrated in *Figure 19: In-RAM representation of radio packet, S0, LENGTH and S1 are optional* on page 82. The PREAMBLE, ADDRESS and CRC fields are omitted in this data structure.

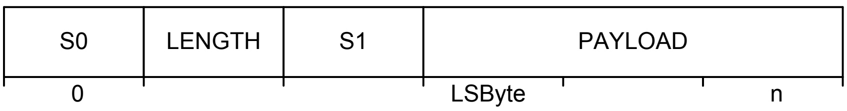



**Figure 19: In-RAM representation of radio packet, S0, LENGTH and S1 are optional**

The byte ordering on air is always Least Significant Byte First for the ADDRESS and PAYLOAD fields and Most Significant Byte First for the CRC field. The ADDRESS fields are always transmitted and received least significant bit first on-air. The CRC field is always transmitted and received Most Significant Bit first. The bit-endian, i.e. which order the bits are sent and received in, of the S0, LENGTH, S1 and PAYLOAD fields can be configured via the ENDIAN in PCNF1.

The sizes, in number of bits, of the S0, LENGTH and S1 fields can be individually configured via S0S, LS and S1S in PCNF0 respectively. If any of these fields are configured to be less than 8 bit long the, the least significant bits of the fields, as seen from the RAM representation, are used.

If S0, LENGTH or S1 are specified with zero length their fields will be omitted in memory, otherwise each field will be represented as a separate byte, regardless of the number of bits in their on-air counterpart.

### 17.1.3 Maximum packet length

Independent of the configuration of MAXLEN, the combined length of S0, LENGTH, S1 and PAYLOAD cannot exceed 254 bytes.

### 17.1.4 Address configuration

The on-air radio ADDRESS field is composed of two parts, the base address field and the address prefix field, see *Table 99: Definition of logical addresses* on page 83. The size of the base address field is configurable via BALEN in PCNF1. The base address is truncated from LSByte if the BALEN is less than 4.

The on-air addresses are defined in the BASEn and PREFIXn registers, and it is only when writing these registers the user will have to relate to actual on-air addresses. For other radio address registers such as the TXADDRESS, RXADDRESSES and RXMATCH registers, logical radio addresses ranging from 0 to 7 are being used. The relationship between the on-air radio addresses and the logical addresses is described in *Table 99: Definition of logical addresses* on page 83.

**Table 99: Definition of logical addresses**

| Logical address | Base address | Prefix byte |
|---|---|---|
| 0 | BASE0 | PREFIX0.AP0 |
| 1 | BASE1 | PREFIX0.AP1 |
| 2 | BASE1 | PREFIX0.AP2 |
| 3 | BASE1 | PREFIX0.AP3 |
| 4 | BASE1 | PREFIX1.AP4 |
| 5 | BASE1 | PREFIX1.AP5 |
| 6 | BASE1 | PREFIX1.AP6 |
| 7 | BASE1 | PREFIX1.AP7 |

## 17.1.5 Received Signal Strength Indicator (RSSI)

The radio implements a mechanism for measuring the power in the received radio signal. This feature is called Received Signal Strength Indicator (RSSI).

Sampling of the received signal strength is started by using the RSSISTART task. The sample can be read from the RSSISAMPLE register.

The sample period of the RSSI is defined by $RSSI_{PERIOD}$, see the device product specification for details. The RSSI sample will hold the average received signal strength during this sample period.

For the RSSI sample to be valid the radio has to be enabled in receive mode (RXEN task) and the reception has to be started (READY event followed by START task).

## 17.1.6 Data whitening

The RADIO is able to do packet whitening and de-whitening, see WHITEEN in PCNF1 register for how to enable whitening. When enabled, whitening and de-whitening will be handled by the RADIO automatically as packets are sent and received, i.e. radio packets located in RAM will not be whitened.

The whitening word is generated using polynomial $g(D) = D^7 + D^4 + 1$, which then is XORed with the data packet that is to be whitened, or de-whitened, see *Figure 20: Data whitening and de-whitening* on page 83.

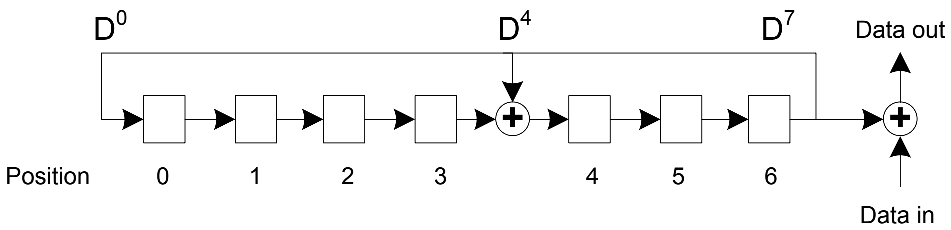



**Figure 20: Data whitening and de-whitening**

Whitening and de-whitening will be performed over the whole packet, except for the preamble, and the address field.

The linear feedback shift register, illustrated in *Figure 20: Data whitening and de-whitening* on page 83 can be initialised via the DATAWHITEIV register.

## 17.1.7 CRC

The CRC generator in the RADIO calculates the CRC over the whole packet excluding the preamble. If desirable the address field can be excluded from the CRC calculation as well, see CRCCNF register for more information.

The CRC polynomial is configurable as illustrated in *Figure 21: CRC generation of an n bit CRC* on page 84 where bit 0 in the CRCPOLY register corresponds to $X^0$ and bit 1 corresponds to $X^1$ etc. See CRCPOLY for more information.

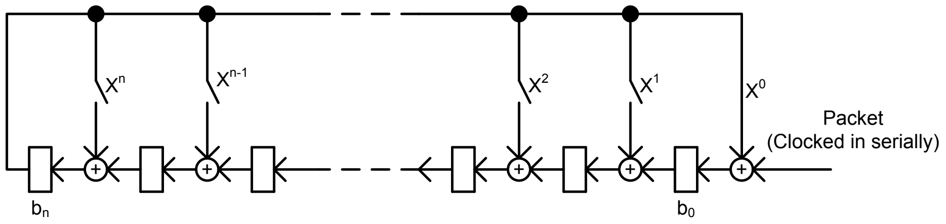



**Figure 21: CRC generation of an n bit CRC**

As illustrated in *Figure 21: CRC generation of an n bit CRC* on page 84, the CRC is calculated by feeding the packet serially through the CRC generator. Before the packet is clocked through the CRC generator, the CRC generator's latches $b_0$ through $b_n$ will be initialized with a predefined value specified in the CRCINIT register. When the whole packet is clocked through the CRC generator, latches $b_0$ through $b_n$ will hold the resulting CRC. This value will be used by the RADIO during both transmission and reception but it is not available to be read by the CPU at any time. A received CRC can however be read by the CPU via the RXCRC register independent of whether or not it has passed the CRC check.

The length (n) of the CRC is configurable, see CRCCNF for more information.

The status of the CRC check can be read from the CRCSTATUS register after a packet has been received.

## 17.1.8 Radio states

The RADIO can enter the following states as described in *Table 100: RADIO state diagram* on page 84 below. An overview state diagram for the RADIO is illustrated in *Figure 22: Radio states* on page 85. This figure shows how the tasks and events relate to the RADIO's operation. The RADIO does not prevent a task from being triggered from the wrong state, if a task is triggered from the wrong state, for example if the RXEN task is triggered from the RXDISABLE state, this may lead to incorrect behaviour. As illustrated in *Figure 22: Radio states* on page 85, the PAYLOAD event is always generated even if the payload is zero.

**Table 100: RADIO state diagram**

| State | Description |
|---|---|
| DISABLED | No operations are going on inside the radio and the power consumption is at a minimum |
| RXRU | The radio is ramping up and preparing for reception |
| RXIDLE | The radio is ready for reception to start |
| RX | Reception has been started and the addresses enabled in the RXADDRESSES register are being monitored |
| TXRU | The radio is ramping up and preparing for transmission |
| TXIDLE | The radio is ready for transmission to start |
| TX | The radio is transmitting a packet |
| RXDISABLE | The radio is disabling the receiver |
| TXDISABLE | The radio is disabling the transmitter |

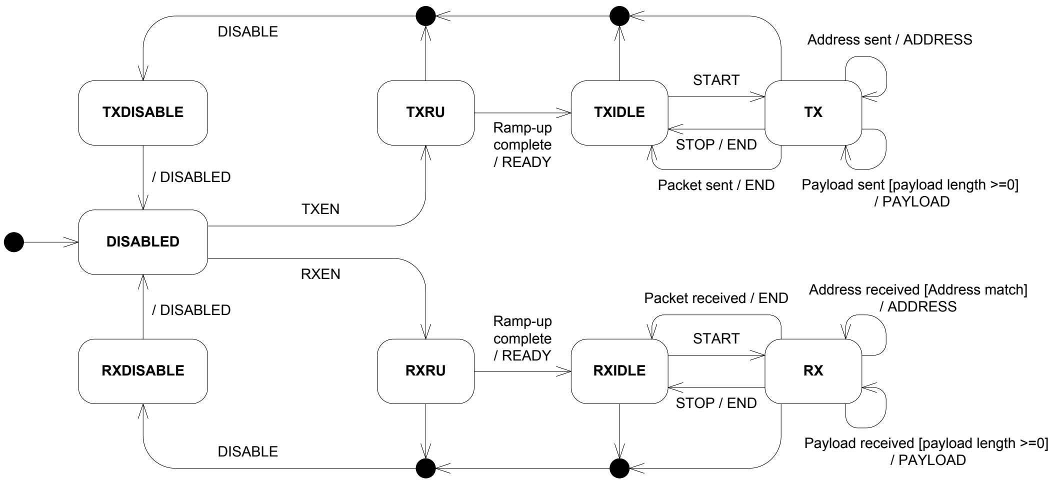



**Figure 22: Radio states**

### 17.1.9 Maximum consecutive transmission time

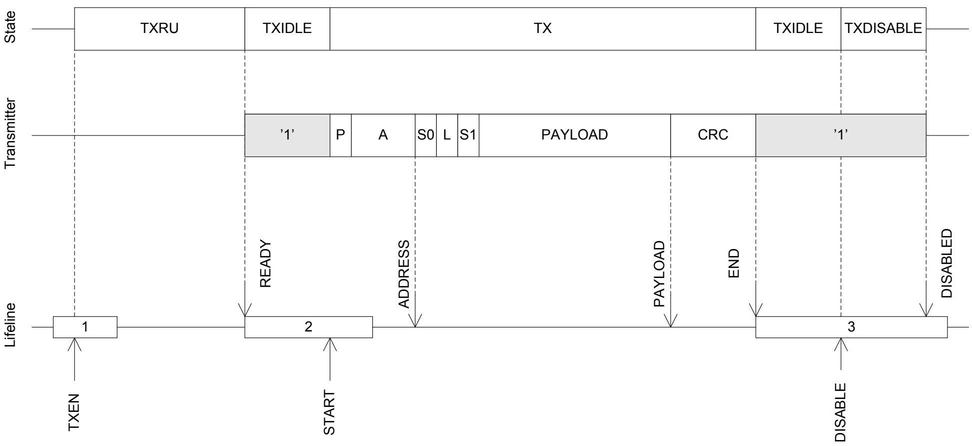
Maximum consecutive transmission time is defined as the longest time the RADIO can be active transmitting before it has to be disabled, i.e. the longest possible time between READY event and DISABLE task.

Maximum consecutive transmission time for the RADIO is 1 ms running of a 60 ppm crystal and 16 ms running of a 30 ppm crystal.

### 17.1.10 Transmit sequence

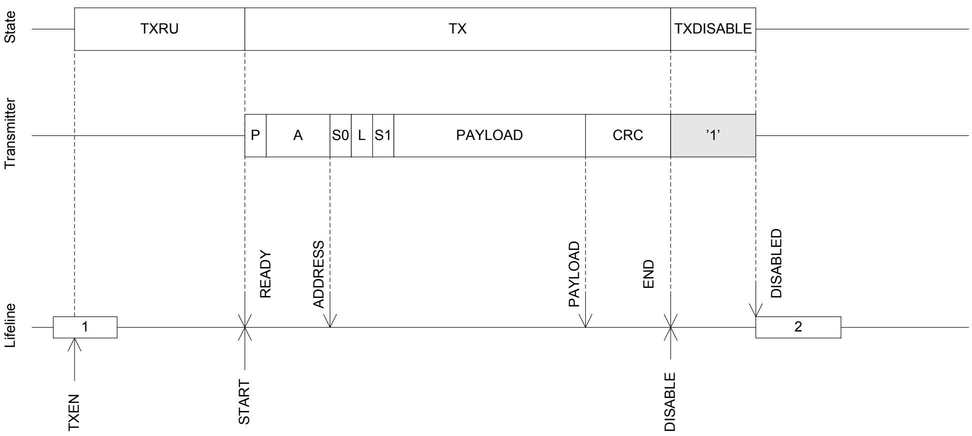
Before the RADIO is able to transmit a packet, it must first ramp-up in TX mode, see TXRU in *Figure 22: Radio states* on page 85 and *Figure 23: Transmit sequence* on page 86 etc. A TXRU ramp-up sequence is initiated when the TXEN task is triggered. After the radio has successfully ramped up it will generate the READY event indicating that a packet transmission can be initiate. A packet transmission is initiated by triggering the START task. As illustrated in *Figure 22: Radio states* on page 85 the START task can first be triggered after the RADIO has entered into the TXIDLE state.

*Figure 23: Transmit sequence* on page 86 illustrates a single packet transmission where the CPU manually triggers the different tasks needed to control the flow of the RADIO, i.e. no shortcuts are used. If shortcuts are not used, a certain amount of delay caused by CPU execution is expected between READY and START, and between END and DISABLE. As illustrated in *Figure 23: Transmit sequence* on page 86 the RADIO will by default transmit '1's between READY and START, and between END and DISABLED.




### Figure 23: Transmit sequence

A slightly modified version of the transmit sequence from *Figure 23: Transmit sequence* on page 86 is illustrated in *Figure 24: Transmit sequence using shortcuts to avoid delays* on page 86 where the RADIO is configured to use shortcuts between READY and START, and between END and DISABLE, which means that no delay is introduced.




### Figure 24: Transmit sequence using shortcuts to avoid delays

The RADIO is able to send multiple packets one after the other without having to disable and re-enable the RADIO between packets, this is illustrated in *Figure 25: Transmission of multiple packets* on page 87.

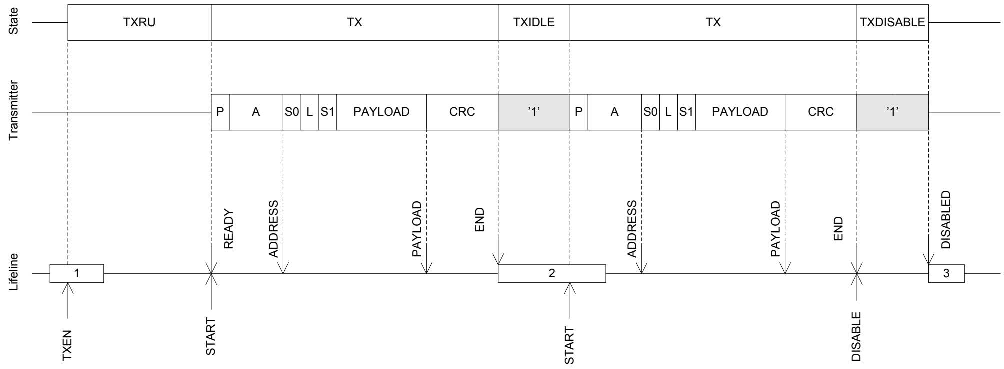



**Figure 25: Transmission of multiple packets**

## 17.1.11 Receive sequence

Before the RADIO is able to receive a packet, it must first ramp-up in RX mode, see RXRU in *Figure 22: Radio states* on page 85 and *Figure 26: Receive sequence* on page 87 etc. An RXRU ramp-up sequence is initiated when the RXEN task is triggered. After the radio has successfully ramped up it will generate the READY event indicating that a packet reception can be initiated. A packet reception is initiated by triggering the START task. As illustrated in *Figure 22: Radio states* on page 85 the START task can, first be triggered after the RADIO has entered into the RXIDLE state.

*Figure 26: Receive sequence* on page 87 illustrates a single packet reception where the CPU manually triggers the different tasks needed to control the flow of the RADIO, i.e. no shortcuts are used. If shortcuts are not used, a certain amount of delay, caused by CPU execution, is expected between READY and START, and between END and DISABLE. As illustrated *Figure 26: Receive sequence* on page 87 the RADIO will be listening and possibly receiving undefined data, illustrated with an 'X', from START and until a packet with valid preamble (P) is received.

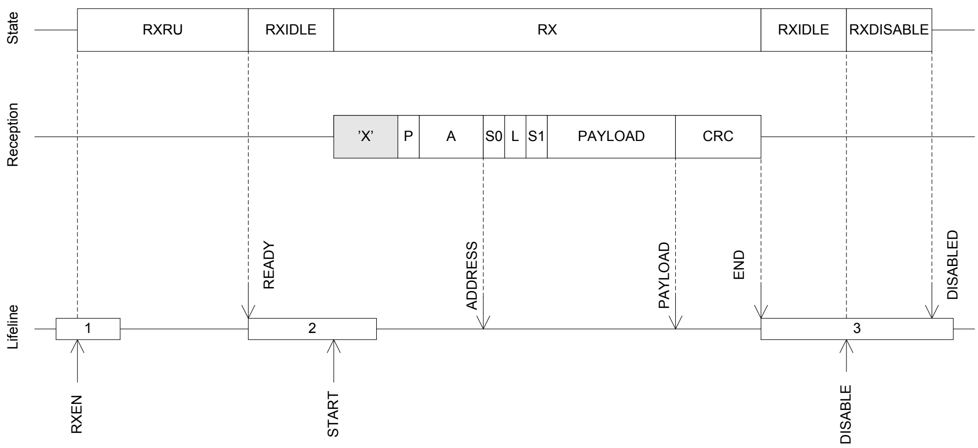



**Figure 26: Receive sequence**

A slightly modified version of the receive sequence from *Figure 26: Receive sequence* on page 87 is illustrated in *Figure 27: Receive sequence using shortcuts to avoid delays* on page 88 where the the RADIO is configured to use shortcuts between READY and START, and between END and DISABLE, which means that no delay is introduced.

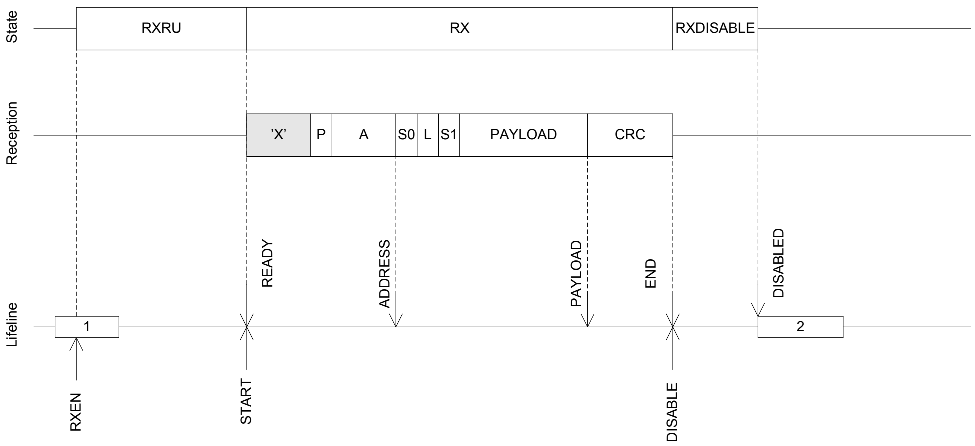



**Figure 27: Receive sequence using shortcuts to avoid delays**

The RADIO is able to receive multiple packets one after the other without having to disable and re-enable the RADIO between packets, this is illustrated *Figure 28: Reception of multiple packets* on page 88.

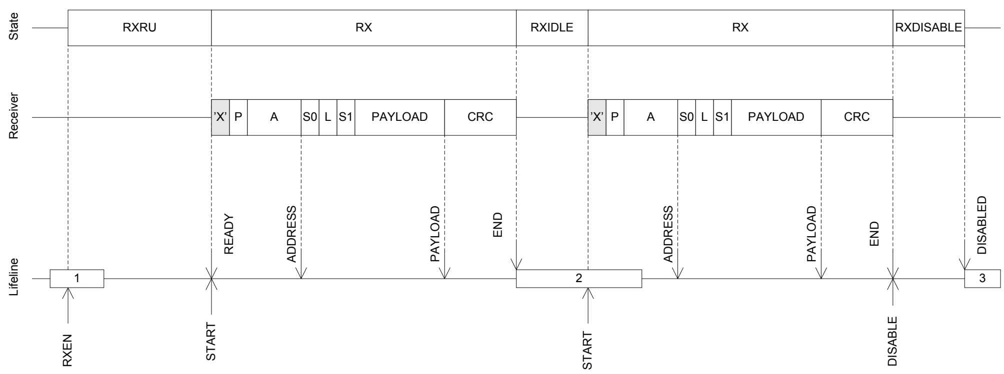



**Figure 28: Reception of multiple packets**

## 17.1.12 Interframe spacing

Interframe spacing is the time interval between two consecutive packets. It is defined as the time, in micro seconds, from the end of the last bit of the previous packet received and to the start of the first bit of the subsequent packet that is transmitted. The RADIO is able to enforce this interval as specified in the TIFS register as long as TIFS is not specified to be shorter than the RADIO’s turn-around time [5], i.e. the time needed to switch off the receiver, and switch back on the transmitter.

TIFS is only enforced if END_DISABLE and DISABLED_TXEN shortcuts are enabled. TIFS is only qualified for use in BLE_1MBIT mode.

## 17.1.13 Device address match

The device address match feature is tailored for address white listing in a Bluetooth Low Energy and similar implementations. This feature enables on-the-fly device address matching while receiving a packet on air. This feature only works in receive mode and as long as RADIO is configured for little endian, see PCNF1.ENDIAN.

[5] See product specification for more information on the timing value $t_{TXEN}$.

The Device Address match unit assumes that the 48 first bits of the payload is the device address and that bit number 6 in S0 is the TxAdd bit. See the Bluetooth Core Specification for more information about device addresses, TxAdd and white listing.

The RADIO is able to listen for 8 different device addresses at the same time. These addresses are specified in a DAB/DAP register pair, one pair per address, in addition to a TxAdd bit configured in the DACNF register. The DAB register specifies the 32 least significant bits of the device address, while the DAP register specifies the 16 most significant bits of the device address.

Each of the device addresses can be individually included or excluded from the matching mechanism. This is configured in the DACNF register.

### 17.1.14 Bit counter

The RADIO implements a simple counter that can be configured to generate an event after a specific number of bits have been transmitted or received. By using shortcuts, this counter can be started from different events generated by the RADIO and hence count relative to these.

The bit counter is started by triggering the BCSTART task, and stopped by triggering the BCSTOP task. A BCMATCH event will be generated when the bit counter has counted the number of bits specified in the BCC register. The bit counter will continue to count bits until the DISABLED event is generated or until the BCSTOP task is triggered. The CPU can therefore, after a BCMATCH event, reconfigure the BCC value for new BCMATCH events within the same packet.

The bit counter can only be started after the RADIO has received the ADDRESS event.

The bit counter will stop and reset on BCSTOP, STOP and DISABLE tasks. The bit counter is also stopped and reset on END event unless the END_START shortcut is enabled.

*Figure 29: Bit counter example* on page 89 illustrate how the bit counter can be used to generate a BCMATCH event in the beginning of the packet payload, and again generate a second BCMATCH event after sending 2 bytes (16 bits) of the payload.

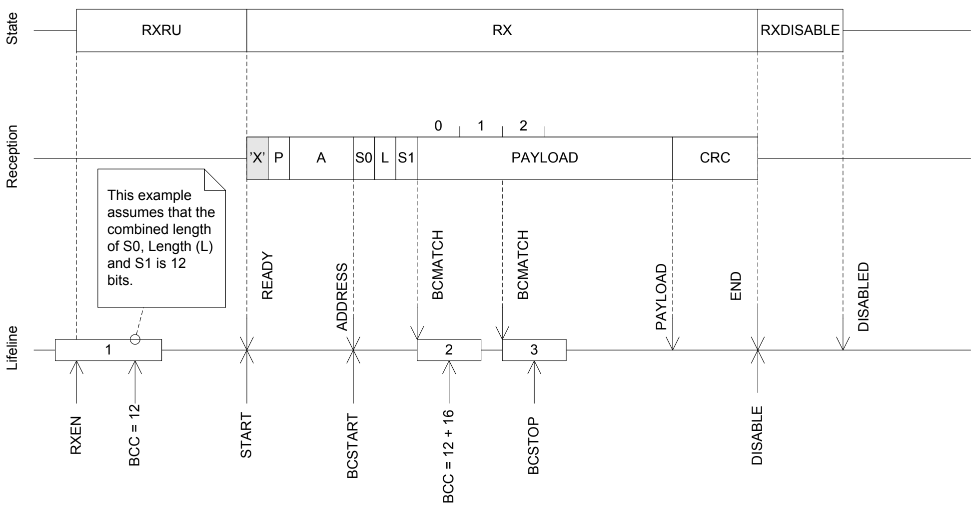



**Figure 29: Bit counter example**

### 17.1.15 Bluetooth trim values

Before the RADIO can be used in BLE_1MBIT mode, see *MODE* register, the default trim values of the RADIO must be overridden if so indicated in the *OVERRIDEEN* register.

See *OVERRIDE0* through *OVERRIDE4* for information about the override registers in the RADIO.

The correct values to specify in the override registers are found in FICR, see *BLE_1MBIT[0]* through *BLE_1MBIT[4]*.

To enable the trim values to be overridden the override mechanism must be enabled via the ENABLE field in the *OVERRIDE4* register. After override is enabled the new trim values will be used next time the RADIO is enabled in TX or RX mode.

To go back to standard trim values, for example when switching between BLE_1MBIT and another RADIO MODE, the override mechanism must be disabled via the ENABLE field in the *OVERRIDE4* register.

## 17.2 Register Overview

### Table 101: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40001000 | RADIO | RADIO | 2.4 GHz Radio |

### Table 102: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *TXEN* | 0x000 | Enable RADIO in TX mode |
| *RXEN* | 0x004 | Enable RADIO in RX mode |
| *START* | 0x008 | Start RADIO |
| *STOP* | 0x00C | Stop RADIO |
| *DISABLE* | 0x010 | Disable RADIO |
| *RSSISTART* | 0x014 | Start the RSSI and take one single sample of the receive signal strength |
| *RSSISTOP* | 0x018 | Stop the RSSI measurement |
| *BCSTART* | 0x01C | Start the bit counter |
| *BCSTOP* | 0x020 | Stop the bit counter |
| Events | | |
| *READY* | 0x100 | RADIO has ramped up and is ready to be started |
| *ADDRESS* | 0x104 | Address sent or received |
| *PAYLOAD* | 0x108 | Packet payload sent or received |
| *END* | 0x10C | Packet sent or received |
| *DISABLED* | 0x110 | RADIO has been disabled |
| *DEVMATCH* | 0x114 | A device address match occurred on the last received packet |
| *DEVMISS* | 0x118 | No device address match occurred on the last received packet |
| *RSSIEND* | 0x11C | Sampling of receive signal strength complete. A new RSSI sample is ready for readout from the *RSSISAMPLE* register. |
| *BCMATCH* | 0x128 | Bit counter reached bit count value specified in the *BCC* register |
| Registers | | |
| *SHORTS* | 0x200 | Shortcut register |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *CRCSTATUS* | 0x400 | CRC status |
| *RXMATCH* | 0x408 | Received address |
| *RXCRC* | 0x40C | CRC field of previously received packet |
| *DAI* | 0x410 | Device address match index |
| *PACKETPTR* | 0x504 | Packet pointer |
| *FREQUENCY* | 0x508 | Frequency |
| *TXPOWER* | 0x50C | Output power |
| *MODE* | 0x510 | Data rate and modulation |
| *PCNF0* | 0x514 | Packet configuration register 0 |
| *PCNF1* | 0x518 | Packet configuration register 1 |
| *BASE0* | 0x51C | Base address 0 |
| *BASE1* | 0x520 | Base address 1 |
| *PREFIX0* | 0x524 | Prefixes bytes for logical addresses 0-3 |
| *PREFIX1* | 0x528 | Prefixes bytes for logical addresses 4-7 |
| *TXADDRESS* | 0x52C | Transmit address select |
| *RXADDRESSES* | 0x530 | Receive address select |
| *CRCCNF* | 0x534 | CRC configuration |
| *CRCPOLY* | 0x538 | CRC polynomial |
| *CRCINIT* | 0x53C | CRC initial value |
| *TEST* | 0x540 | Test features enable register. |
| *TIFS* | 0x544 | Inter Frame Spacing in us |
| *RSSISAMPLE* | 0x548 | RSSI sample |
| *STATE* | 0x550 | Current radio state |
| *DATAWHITEIV* | 0x554 | Data whitening initial value |
| *BCC* | 0x560 | Bit counter compare |
| *DAB[0]* | 0x600 | Device address base segment 0 |
| *DAB[1]* | 0x604 | Device address base segment 1 |
| *DAB[2]* | 0x608 | Device address base segment 2 |
| *DAB[3]* | 0x60C | Device address base segment 3 |
| *DAB[4]* | 0x610 | Device address base segment 4 |
| *DAB[5]* | 0x614 | Device address base segment 5 |
| *DAB[6]* | 0x618 | Device address base segment 6 |
| *DAB[7]* | 0x61C | Device address base segment 7 |
| *DAP[0]* | 0x620 | Device address prefix 0 |
| *DAP[1]* | 0x624 | Device address prefix 1 |
| *DAP[2]* | 0x628 | Device address prefix 2 |

| Register | Offset | Description |
|---|---|---|
| *DAP[3]* | 0x62C | Device address prefix 3 |
| *DAP[4]* | 0x630 | Device address prefix 4 |
| *DAP[5]* | 0x634 | Device address prefix 5 |
| *DAP[6]* | 0x638 | Device address prefix 6 |
| *DAP[7]* | 0x63C | Device address prefix 7 |
| *DACNF* | 0x640 | Device address match configuration |
| *OVERRIDE0* | 0x724 | Trim value override register 0 |
| *OVERRIDE1* | 0x728 | Trim value override register 1 |
| *OVERRIDE2* | 0x72C | Trim value override register 2 |
| *OVERRIDE3* | 0x730 | Trim value override register 3 |
| *OVERRIDE4* | 0x734 | Trim value override register 4 |
| *POWER* | 0xFFC | Peripheral power control |

# 17.3 Register Details

## Table 103: SHORTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY_START | | | Shortcut between *READY* event and *START* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| B | RW | END_DISABLE | | | Shortcut between *END* event and *DISABLE* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| C | RW | DISABLED_TXEN | | | Shortcut between *DISABLED* event and *TXEN* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| D | RW | DISABLED_RXEN | | | Shortcut between *DISABLED* event and *RXEN* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| E | RW | ADDRESS_RSSISTART | | | Shortcut between *ADDRESS* event and *RSSISTART* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| F | RW | END_START | | | Shortcut between *END* event and *START* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| G | RW | ADDRESS_BCSTART | | | Shortcut between *ADDRESS* event and *BCSTART* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| H | RW | DISABLED_RSSISTOP | | | Shortcut between *DISABLED* event and *RSSISTOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

## Table 104: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | | | | | | | | | | | | | | | | | | | | | | K | | | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY | | | Write '1' to Enable interrupt on *READY* event. |
| | | | Enabled | 1 | Enable |
| B | RW | ADDRESS | | | Write '1' to Enable interrupt on *ADDRESS* event. |
| | | | Enabled | 1 | Enable |
| C | RW | PAYLOAD | | | Write '1' to Enable interrupt on *PAYLOAD* event. |
| | | | Enabled | 1 | Enable |
| D | RW | END | | | Write '1' to Enable interrupt on *END* event. |
| | | | Enabled | 1 | Enable |
| E | RW | DISABLED | | | Write '1' to Enable interrupt on *DISABLED* event. |
| | | | Enabled | 1 | Enable |
| F | RW | DEVMATCH | | | Write '1' to Enable interrupt on *DEVMATCH* event. |
| | | | Enabled | 1 | Enable |
| G | RW | DEVMISS | | | Write '1' to Enable interrupt on *DEVMISS* event. |
| | | | Enabled | 1 | Enable |
| H | RW | RSSIEND | | | Write '1' to Enable interrupt on *RSSIEND* event. |
| | | | Enabled | 1 | Enable |
| K | RW | BCMATCH | | | Write '1' to Enable interrupt on *BCMATCH* event. |
| | | | Enabled | 1 | Enable |

## Table 105: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | | | | | | | | | | | | | | | | | | | | | | K | | | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY | | | Write '1' to Clear interrupt on *READY* event. |
| | | | Disabled | 1 | Disable |
| B | RW | ADDRESS | | | Write '1' to Clear interrupt on *ADDRESS* event. |
| | | | Disabled | 1 | Disable |
| C | RW | PAYLOAD | | | Write '1' to Clear interrupt on *PAYLOAD* event. |
| | | | Disabled | 1 | Disable |
| D | RW | END | | | Write '1' to Clear interrupt on *END* event. |
| | | | Disabled | 1 | Disable |
| E | RW | DISABLED | | | Write '1' to Clear interrupt on *DISABLED* event. |
| | | | Disabled | 1 | Disable |
| F | RW | DEVMATCH | | | Write '1' to Clear interrupt on *DEVMATCH* event. |
| | | | Disabled | 1 | Disable |
| G | RW | DEVMISS | | | Write '1' to Clear interrupt on *DEVMISS* event. |
| | | | Disabled | 1 | Disable |
| H | RW | RSSIEND | | | Write '1' to Clear interrupt on *RSSIEND* event. |
| | | | Disabled | 1 | Disable |
| K | RW | BCMATCH | | | Write '1' to Clear interrupt on *BCMATCH* event. |
| | | | Disabled | 1 | Disable |

## Table 106: CRCSTATUS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | CRCSTATUS | | | CRC status of packet received |
| | | | CRCError | 0 | Packet received with CRC error |
| | | | CRCOk | 1 | Packet received with CRC ok |

## Table 107: RXMATCH

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RXMATCH | | | Received address<br>Logical address of which previous packet was received |

## Table 108: RXCRC

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RXCRC | | | CRC field of previously received packet<br>CRC field of previously received packet |

## Table 109: DAI

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | DAI | | | Device address match index<br>Index (n) of device address, see *DAB[n]* and *DAP[n]*, that got an address match. |

## Table 110: PACKETPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PACKETPTR | | | Packet pointer<br>Packet address to be used for the next transmission or reception. When transmitting, the packet pointed to by this address will be transmitted and when receiving, the received packet will be written to this address. This address is a byte aligned ram address.<br>**Attention:** Decision point: *START* |

## Table 111: 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | FREQUENCY | | [0..100] | Radio channel frequency<br>Frequency = 2400 + FREQUENCY (MHz).<br>**Attention:** Decision point: *TXEN*, *RXEN* |

## Table 112: TXPOWER

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TXPOWER | | | RADIO output power.<br>Output power in number of dBm, i.e. if the value -20 is specified the output power will be set to -20dBm.<br>**Attention:** Decision point: *TXEN* |
| | | | Pos4dBm | 0x04 | +4 dBm |
| | | | 0dBm | 0x00 | 0 dBm |
| | | | Neg4dBm | 0xFC | -4 dBm |
| | | | Neg8dBm | 0xF8 | -8 dBm |
| | | | Neg12dBm | 0xF4 | -12 dBm |
| | | | Neg16dBm | 0xF0 | -16 dBm |
| | | | Neg20dBm | 0xEC | -20 dBm |
| | | | Neg30dBm | 0xD8 | -30 dBm |

## Table 113: 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | MODE | | | Radio data rate and modulation setting. The radio supports Frequency-shift Keying (FSK) modulation. |
| | | | Nrf_1Mbit | 0 | 1 Mbit/s Nordic proprietary radio mode |
| | | | Nrf_2Mbit | 1 | 2 Mbit/s Nordic proprietary radio mode |
| | | | Nrf_250Kbit | 2 | 250 kbit/s Nordic proprietary radio mode |
| | | | Ble_1Mbit | 3 | 1 Mbit/s Bluetooth Low Energy |

## Table 114: PCNF0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | E | E | | | | | | | | C | | | | | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | LFLEN | | | Length on air of LENGTH field in number of bits.<br>**Attention:** Decision point: *START* |
| C | RW | S0LEN | | | Length on air of S0 field in number of bytes.<br>**Attention:** Decision point: *START* |
| E | RW | S1LEN | | | Length on air of S1 field in number of bits.<br>**Attention:** Decision point: *START* |

## Table 115: PCNF1

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | C | C | B | B | B | B | B | B | B | B | A | A | A | A | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | MAXLEN | | [0..255] | Maximum length of packet payload. If the packet payload is larger than MAXLEN, the radio will truncate the payload to MAXLEN. |
| B | RW | STATLEN | | [0..255] | Static length in number of bytes<br>The static length parameter is added to the total length of the payload when sending and receiving packets, e.g. if the static length is set to N the radio will receive or send N bytes more than what is defined in the LENGTH field of the packet.<br>**Attention:** Decision point: *START* |
| C | RW | BALEN | | [2..4] | Base address length in number of bytes<br>The address field is composed of the base address and the one byte long address prefix, e.g. set BALEN=2 to get a total address of 3 bytes.<br>**Attention:** Decision point: *START* |
| D | RW | ENDIAN | | | On air endianness of packet, this applies to the S0, LENGTH, S1 and the PAYLOAD fields.<br>**Attention:** Decision point: *START* |
| | | | Little | 0 | Least Significant bit on air first |
| | | | Big | 1 | Most significant bit on air first |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | C | C | B | B | B | B | B | B | B | B | A | A | A | A | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| E | RW | WHITEEN | | | Enable or disable packet whitening |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

## Table 116: BASE0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | BASE0 | | | Base address 0<br>Radio base address 0.<br>**Attention:** Decision point: *START* |

## Table 117: BASE1

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | BASE1 | | | Base address 1<br>Radio base address 1.<br>**Attention:** Decision point: *START* |

## Table 118: PREFIX0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | D | D | D | D | C | C | C | C | C | C | C | C | B | B | B | B | B | B | B | B | A | A | A | A | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | AP0 | | | Address prefix 0.<br>**Attention:** Decision point: *START* |
| B | RW | AP1 | | | Address prefix 1.<br>**Attention:** Decision point: *START* |
| C | RW | AP2 | | | Address prefix 2.<br>**Attention:** Decision point: *START* |
| D | RW | AP3 | | | Address prefix 3.<br>**Attention:** Decision point: *START* |

## Table 119: PREFIX1

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | D | D | D | D | C | C | C | C | C | C | C | C | B | B | B | B | B | B | B | B | A | A | A | A | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | AP4 | | | Address prefix 4.<br>**Attention:** Decision point: *START* |
| B | RW | AP5 | | | Address prefix 5.<br>**Attention:** Decision point: *START* |
| C | RW | AP6 | | | Address prefix 6.<br>**Attention:** Decision point: *START* |
| D | RW | AP7 | | | Address prefix 7.<br>**Attention:** Decision point: *START* |

## Table 120: TXADDRESS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TXADDRESS | | | Transmit address select<br>Logical address to be used when transmitting a packet.<br>**Attention:** Decision point: *START* |

## Table 121: RXADDRESSES

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ADDR0 | | | Enable or disable reception on logical address 0.<br>**Attention:** Decision point: *START* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | ADDR1 | | | Enable or disable reception on logical address 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 | | | | | | | | | | | | | | | | | | | | | | | | | H | G | F | E | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | | | **Attention:** Decision point: *START* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| C | RW | ADDR2 | | | Enable or disable reception on logical address 2. |
| | | | | | **Attention:** Decision point: *START* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| D | RW | ADDR3 | | | Enable or disable reception on logical address 3. |
| | | | | | **Attention:** Decision point: *START* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| E | RW | ADDR4 | | | Enable or disable reception on logical address 4. |
| | | | | | **Attention:** Decision point: *START* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| F | RW | ADDR5 | | | Enable or disable reception on logical address 5. |
| | | | | | **Attention:** Decision point: *START* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| G | RW | ADDR6 | | | Enable or disable reception on logical address 6. |
| | | | | | **Attention:** Decision point: *START* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| H | RW | ADDR7 | | | Enable or disable reception on logical address 7. |
| | | | | | **Attention:** Decision point: *START* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

## Table 122: CRCCNF

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | LEN | | [1..3] | CRC length in number of bytes. |
| | | | | | **Attention:** Decision point: *START* |
| | | | Disabled | 0 | CRC length is zero and CRC calculation is disabled |
| | | | One | 1 | CRC length is one byte and CRC calculation is enabled |
| | | | Two | 2 | CRC length is two bytes and CRC calculation is enabled |
| | | | Three | 3 | CRC length is three bytes and CRC calculation is enabled |
| B | RW | SKIPADDR | | | Include or exclude packet address field out of CRC calculation. |
| | | | | | **Attention:** Decision point: *START* |
| | | | Include | 0 | CRC calculation includes address field |
| | | | Skip | 1 | CRC calculation does not include address field. The CRC calculation will start at the first byte after the address. |

## Table 123: CRCPOLY

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CRCPOLY | | | CRC polynomial<br>Each term in the CRC polynomial is mapped to a bit in this register which index corresponds to the term's exponent. The least significant term/bit is hard-wired internally to 1, and bit number 0 of the register content is ignored by the hardware. The following example is for an 8 bit CRC polynomial: x8 + x7 + x3 + x2 + 1 = 1 1000 1101 . |
| | | | | | **Attention:** Decision point: *START* |

## Table 124: CRCINIT

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CRCINIT | | | CRC initial value<br>Initial value for CRC calculation. |
| | | | | | **Attention:** Decision point: *START* |

### Table 125: TEST

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CONSTCARRIER | | | Enable or disable constant carrier.<br>**Attention:** Decision point: *TXEN* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | PLLLOCK | | | Enable or disable PLL lock.<br>**Attention:** Decision point: *TXEN*, *RXEN* |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 126: TIFS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TIFS | | | Inter Frame Spacing in us<br>Inter frame space is the time interval between two consecutive packets. It is defined as the time, in micro seconds, from the end of the last bit of the previous packet to the start of the first bit of the subsequent packet.<br>**Attention:** Decision point: *START* |

### Table 127: RSSISAMPLE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RSSISAMPLE | | [0..127] | RSSI sample<br>RSSI sample result. The value of this register is read as a positive value while the actual received signal strength is a negative value. Actual received signal strength is therefore as follows: received signal strength = -A dBm |

### Table 128: 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | STATE | | | Current radio state |
| | | | Disabled | 0 | RADIO is in the Disabled state |
| | | | RxRu | 1 | RADIO is in the RXRU state |
| | | | RxIdle | 2 | RADIO is in the RXIDLE state |
| | | | Rx | 3 | RADIO is in the RX state |
| | | | RxDisable | 4 | RADIO is in the RXDISABLED state |
| | | | TxRu | 9 | RADIO is in the TXRU state |
| | | | TxIdle | 10 | RADIO is in the TXIDLE state |
| | | | Tx | 11 | RADIO is in the TX state |
| | | | TxDisable | 12 | RADIO is in the TXDISABLED state |

### Table 129: DATAWHITEIV

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 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 | 0 | 0 | 0 | 0 | 0 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | DATAWHITEIV | | | Data whitening initial value. Bit 6 is hard-wired to '1', writing '0' to it has no effect, and it will always be read back and used by the device as '1'.<br>Bit 0 corresponds to Position 6 of the LSFR, Bit 1 to Position 5, etc.<br>**Attention:** Decision point: *TXEN*, *RXEN* |

### Table 130: BCC

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | BCC | | | Bit counter compare<br>Bit counter compare register |

## Table 131: DAB[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 | A | A | A | A | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DAB | | | Device address base segment n<br>Device address base segment |

## Table 132: DAP[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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DAP | | | Device address prefix n<br>Device address prefix |

## Table 133: DACNF

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENA0 | | | Enable or disable device address matching using device address 0 |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| B | RW | ENA1 | | | Enable or disable device address matching using device address 1 |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| C | RW | ENA2 | | | Enable or disable device address matching using device address 2 |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| D | RW | ENA3 | | | Enable or disable device address matching using device address 3 |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| E | RW | ENA4 | | | Enable or disable device address matching using device address 4 |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| F | RW | ENA5 | | | Enable or disable device address matching using device address 5 |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| G | RW | ENA6 | | | Enable or disable device address matching using device address 6 |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| H | RW | ENA7 | | | Enable or disable device address matching using device address 7 |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| I | RW | TXADD0 | | | TxAdd for device address 0 |
| J | RW | TXADD1 | | | TxAdd for device address 1 |
| K | RW | TXADD2 | | | TxAdd for device address 2 |
| L | RW | TXADD3 | | | TxAdd for device address 3 |
| M | RW | TXADD4 | | | TxAdd for device address 4 |
| N | RW | TXADD5 | | | TxAdd for device address 5 |
| O | RW | TXADD6 | | | TxAdd for device address 6 |
| P | RW | TXADD7 | | | TxAdd for device address 7 |

## Table 134: OVERRIDE0

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OVERRIDE0 | | | Trim value override register 0 |

## Table 135: OVERRIDE1

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OVERRIDE1 | | | Trim value override register 1 |

## Table 136: OVERRIDE2

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OVERRIDE2 | | | Trim value override register 2 |

## Table 137: OVERRIDE3

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OVERRIDE3 | | | Trim value override register 3 |

## Table 138: OVERRIDE4

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | A | A | A | A | A | A | A | A | A | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OVERRIDE4 | | | Trim value override register 4 |
| B | RW | ENABLE | | | Enable or disable override of default trim values |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

## Table 139: POWER

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | POWER | | | Peripheral power control. The peripheral and its registers will be reset to its initial state by switching the peripheral off and then back on again. |
| | | | Disabled | 0 | Peripheral is powered off |
| | | | Enabled | 1 | Peripheral is powered on |

# 18 Timer/counter (TIMER)

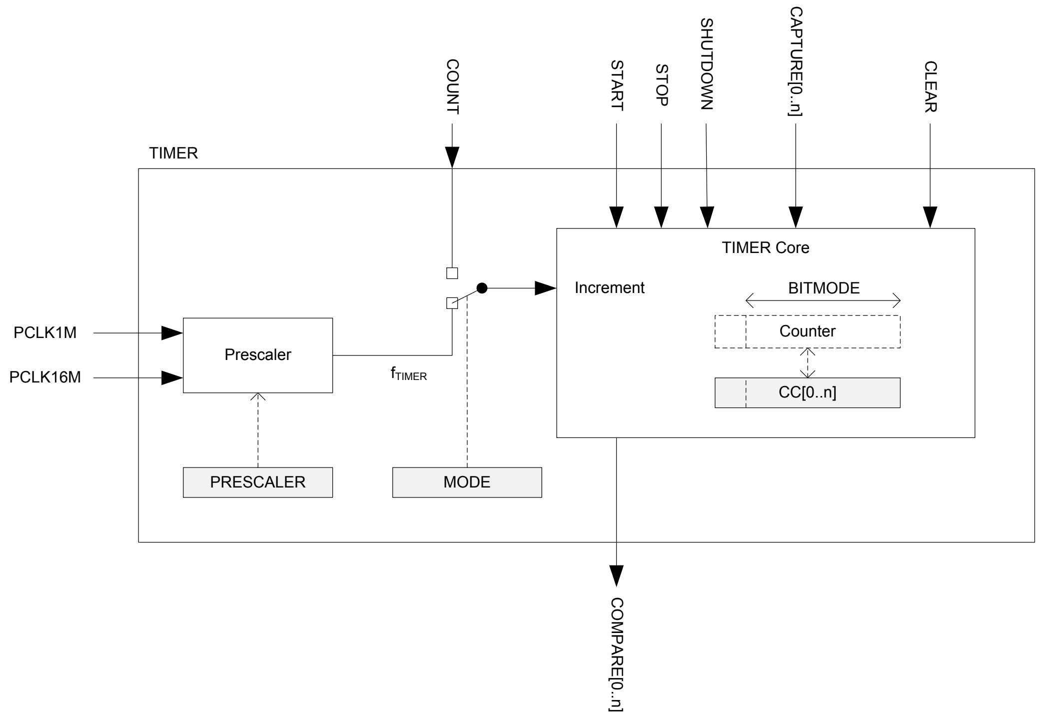



**Figure 30: Block schematic for timer/counter**

## 18.1 Functional description

The TIMER can operate in two modes, Timer mode and Counter mode. In both modes the TIMER is started by triggering the START task, and stopped by triggering the STOP task. After the timer is stopped the timer can resume timing/counting by triggering the START task again. When timing/counting is resumed the timer will continue from the value it had prior to being stopped.

If the timer does not need to be able to resume timing/counting after a STOP the SHUTDOWN task could be used instead of or following the STOP task.

When the timer is shut down the internal core of the timer, as illustrated in *Figure 30: Block schematic for timer/counter* on page 99, is switched off. To reach lowest power consumption in system ON mode the timer must be shut down. The startup time from shutdown state may be longer compared to starting the timer from the stopped state. See *Power management (POWER)* on page 42 for more information about power modes. See product specification for more information about startup times and power consumption.

In Timer mode, the TIMER's internal Counter register is incremented by one for every tick of the timer frequency $f_{TIMER}$ as illustrated in *Figure 30: Block schematic for timer/counter* on page 99. The timer frequency is derived from PCLK16M as described in *Equation 1* using the values specified in the PRESCALER register:

$$f_{TIMER} = 16\ \text{MHz} / (2^{PRESCALER})$$

When $f_{TIMER}$ <= 1 MHz the TIMER will use PCLK1M instead of PCLK16M for reduced power consumption.

In counter mode, the TIMER's internal Counter register is incremented by one each time the COUNT task is triggered, that is, the timer frequency and the prescaler are not utilized in counter mode. Similarly, the COUNT task has no effect in Timer mode.

The TIMER's maximum value is configured by changing the bit-width of the timer in the BITMODE register. For details on which bitmodes are supporting which timers see the device product specification.

The PRESCALER register and the BITMODE register must only be updated when the timer is stopped. If these registers are updated while the TIMER is started then this may result in unpredictable behavior.

When the timer is incremented beyond its maximum value the Counter register will overflow and the TIMER will automatically start over from zero.

The Counter register can be cleared, that is, its internal value set to zero explicitly, by triggering the CLEAR task.

The TIMER implements multiple capture/compare registers, see the product specification for more information on how many capture/compare registers that are supported in the chip.

### 18.1.1 Capture

The TIMER implements one capture task for every available capture/compare register. Every time the CAPTURE[n] task is triggered the Counter value is copied to the CC[n] register.

### 18.1.2 Compare

The TIMER implements one COMPARE event for every available capture/compare register. A COMPARE event is generated when the Counter is incremented and then becomes equal to the value specified in one of the capture compare registers. When the Counter value becomes equal to the value specified in a capture compare register CC[n], the corresponding compare event COMPARE[n] is generated. The amount of compare registers per TIMER instantiation is defined in the Product Specification.

BITMODE specifies how many bits of the Counter register and the capture/compare register that are used when the comparison is performed. Other bits will be ignored.

### 18.1.3 Task delays

The CLEAR task, COUNT task and the STOP task will guarantee to take effect within one clock cycle of the PCLK16M. Depending on sub-power mode, the START task may require longer time to take effect, see product specification for more information. See *POWER* chapter for more information about sub-power modes.

### 18.1.4 Task priority

If the START task and the STOP task are triggered at the same time, that is, within the same period of PCLK16M, the STOP task will be prioritized.

## 18.2 Register Overview

#### Table 140: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40008000 | TIMER | TIMER0 | Timer/Counter |
| 0x40009000 | TIMER | TIMER1 | Timer/Counter |
| 0x4000A000 | TIMER | TIMER2 | Timer/Counter |

#### Table 141: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *START* | 0x000 | Start Timer |
| *STOP* | 0x004 | Stop Timer |
| *COUNT* | 0x008 | Increment Timer (Counter mode only) |
| *CLEAR* | 0x00C | Clear time |
| *SHUTDOWN* | 0x010 | Shut down timer |
| *CAPTURE[0]* | 0x040 | Capture Timer value to CC[0] register |
| *CAPTURE[1]* | 0x044 | Capture Timer value to CC[1] register |
| *CAPTURE[2]* | 0x048 | Capture Timer value to CC[2] register |
| *CAPTURE[3]* | 0x04C | Capture Timer value to CC[3] register |

| Register | Offset | Description |
|---|---|---|
| Events | | |
| *COMPARE[0]* | 0x140 | Compare event on CC[0] match |
| *COMPARE[1]* | 0x144 | Compare event on CC[1] match |
| *COMPARE[2]* | 0x148 | Compare event on CC[2] match |
| *COMPARE[3]* | 0x14C | Compare event on CC[3] match |
| Registers | | |
| *SHORTS* | 0x200 | Shortcut register |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *MODE* | 0x504 | Timer mode selection |
| *BITMODE* | 0x508 | Configure the number of bits used by the TIMER |
| *PRESCALER* | 0x510 | Timer prescaler register |
| *CC[0]* | 0x540 | Capture/Compare register 0 |
| *CC[1]* | 0x544 | Capture/Compare register 1 |
| *CC[2]* | 0x548 | Capture/Compare register 2 |
| *CC[3]* | 0x54C | Capture/Compare register 3 |

# 18.3 Register Details

## Table 142: SHORTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | | | | | | | | | | | | | | | | | | | | | J | I | H | G | | | | | D | C | B | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | COMPARE0_CLEAR | | | Shortcut between *COMPARE[0]* event and *CLEAR* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| B | RW | COMPARE1_CLEAR | | | Shortcut between *COMPARE[1]* event and *CLEAR* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| C | RW | COMPARE2_CLEAR | | | Shortcut between *COMPARE[2]* event and *CLEAR* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| D | RW | COMPARE3_CLEAR | | | Shortcut between *COMPARE[3]* event and *CLEAR* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| G | RW | COMPARE0_STOP | | | Shortcut between *COMPARE[0]* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| H | RW | COMPARE1_STOP | | | Shortcut between *COMPARE[1]* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| I | RW | COMPARE2_STOP | | | Shortcut between *COMPARE[2]* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| J | RW | COMPARE3_STOP | | | Shortcut between *COMPARE[3]* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

## Table 143: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | COMPARE0 | | | Write '1' to Enable interrupt on *COMPARE[0]* event. |
| | | | Enabled | 1 | Enable |
| B | RW | COMPARE1 | | | Write '1' to Enable interrupt on *COMPARE[1]* event. |
| | | | Enabled | 1 | Enable |
| C | RW | COMPARE2 | | | Write '1' to Enable interrupt on *COMPARE[2]* event. |
| | | | Enabled | 1 | Enable |
| D | RW | COMPARE3 | | | Write '1' to Enable interrupt on *COMPARE[3]* event. |
| | | | Enabled | 1 | Enable |

## Table 144: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | COMPARE0 | | | Write '1' to Clear interrupt on *COMPARE[0]* event. |
| | | | Disabled | 1 | Disable |
| B | RW | COMPARE1 | | | Write '1' to Clear interrupt on *COMPARE[1]* event. |
| | | | Disabled | 1 | Disable |
| C | RW | COMPARE2 | | | Write '1' to Clear interrupt on *COMPARE[2]* event. |

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Disabled | 1 | Disable |
| D | RW | COMPARE3 | | | Write '1' to Clear interrupt on *COMPARE[3]* event. |
| | | | Disabled | 1 | Disable |

## Table 145: 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | MODE | | | Timer mode |
| | | | Timer | 0 | Select Timer mode |
| | | | Counter | 1 | Select Counter mode |

## Table 146: BITMODE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | BITMODE | | | Timer bit width |
| | | | 16Bit | 0 | 16 bit timer bit width |
| | | | 08Bit | 1 | 8 bit timer bit width |
| | | | 24Bit | 2 | 24 bit timer bit width |
| | | | 32Bit | 3 | 32 bit timer bit width |

## Table 147: PRESCALER

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 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 | 0 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PRESCALER | | [0..9] | Prescaler value |

## Table 148: CC[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 | A | A | A | A | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CC | | | Capture/Compare value<br>Only the number of bits indicated by BITMODE will be used by the TIMER. |

# 19 Real Time Counter (RTC)

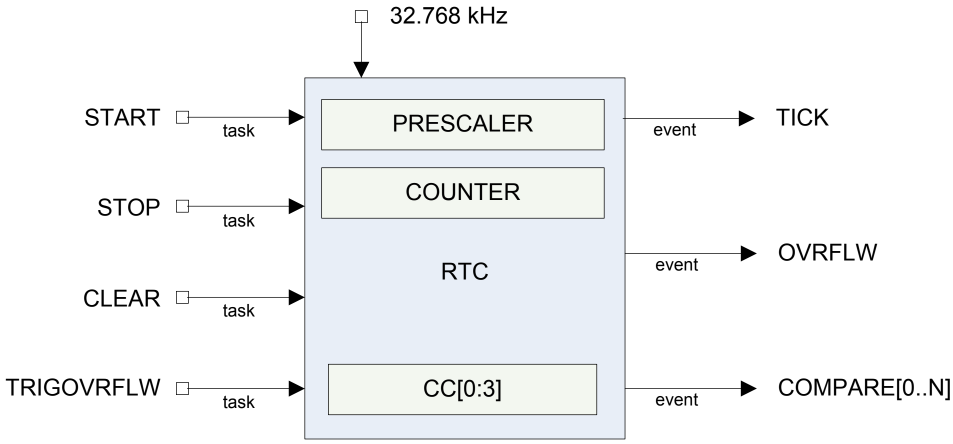



**Figure 31: RTC block schematic**

## 19.1 Functional description

The RTC is a 24 bit low-frequency clock with frequency prescaling and tick, compare, and overflow events.

### 19.1.1 Clock source

The RTC will run off the LFCLK, see *Clock management (CLOCK)* on page 51 for more information about clock sources. The COUNTER resolution will therefore be 30.517 µs. The RTC is able to run while the HFCLK is OFF and PCLK16M is not available.

### 19.1.2 Resolution versus overflow and the PRESCALER

Counter increment frequency:

$$f_{RTC}\ [kHz] = 32.768\ /\ (PRESCALER + 1\ )$$

The PRESCALER register is read/write when the RTC is stopped. The RESCALER register is read-only once the RTC is STARTed. Writing to the RESCALER register when the RTC is started has no effect.

The PRESCALER is restarted on 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

**Table 149: RTC resolution versus overflow**

| 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 |

## 19.1.3 The COUNTER register

The COUNTER increments on LFCLK when the internal PRESCALER register (<<PRESC>>) is 0x00. The internal <<PRESC>> register is reloaded from the PRESCALER register. If enabled, the TICK event occurs on each increment of the COUNTER. The TICK event can be disabled.

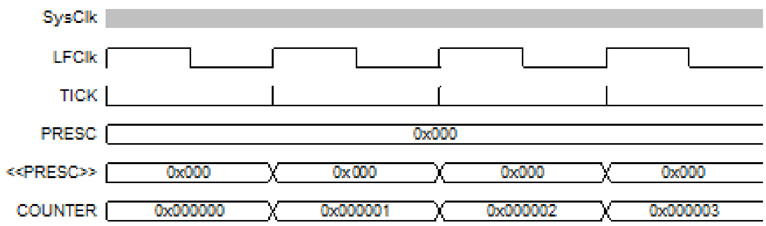



**Figure 32: Timing diagram - COUNTER_PRESCALER_0**

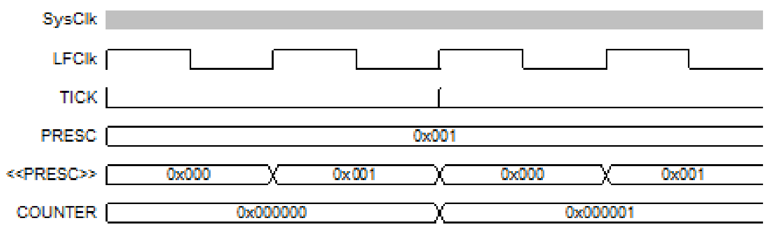



**Figure 33: Timing diagram - COUNTER_PRESCALER_1**

## 19.1.4 Overflow features

The TRIGOVRFLW task sets the COUNTER value to 0xFFFFF0 to allow SW test of the overflow condition. OVRFLW occurs when COUNTER overflows from 0xFFFFFF to 0x000000.

**Note:**

The OVRFLW event is disabled by default.

## 19.1.5 The TICK event

The TICK event enables low power RTOS implementation as it optionally provides a regular interrupt source for a RTOS without the need to use the ARM® SysTick feature. Using the RTC 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.

## 19.1.6 Event Control feature

To optimize RTC power consumption, events in the RTC can be individually disabled to prevent PCLK16M and HFCLK 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, this event should be disabled as it is frequently occurring and may raise power consumption if HFCLK otherwise could be powered down for long durations.

This means that the RTC implements a slightly different task and event system compared to the standard system described in *Figure 6: Tasks, events, shortcuts, and interrupts* on page 37. The RTC's task and event system is illustrated in *Figure 34: Tasks, events and interrupts in the RTC* on page 105.

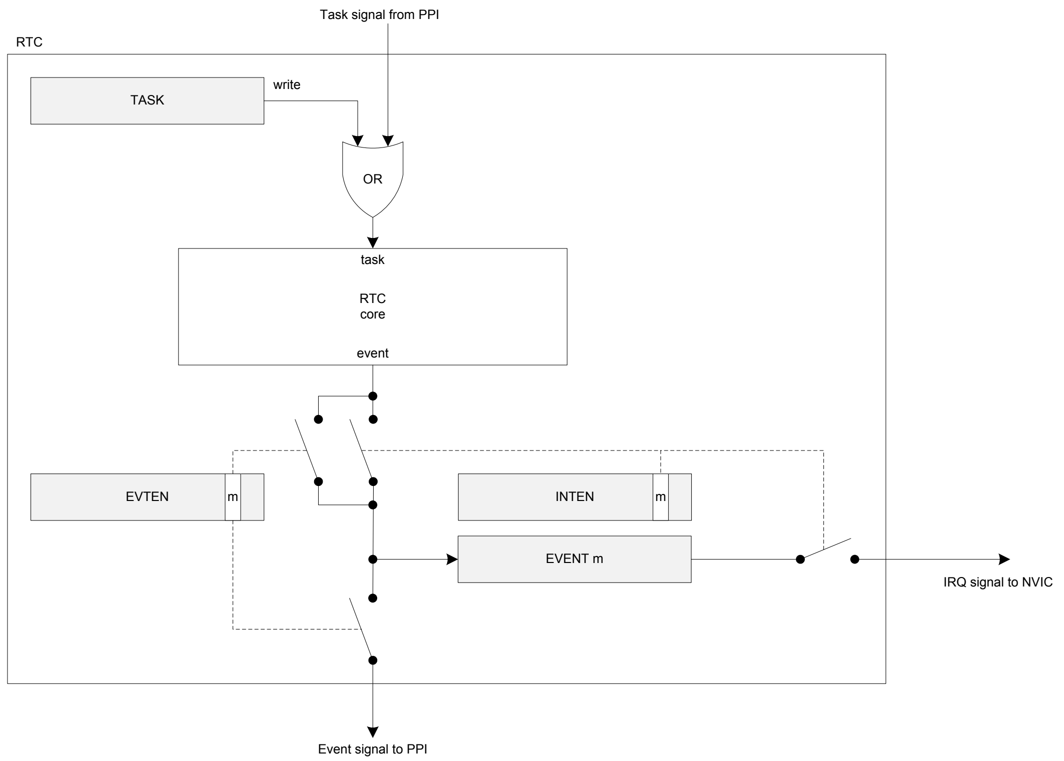



**Figure 34: Tasks, events and interrupts in the RTC**

## 19.1.7 Compare feature

There are three supported compare registers and one optional. See product specification for details on available compare registers.

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.

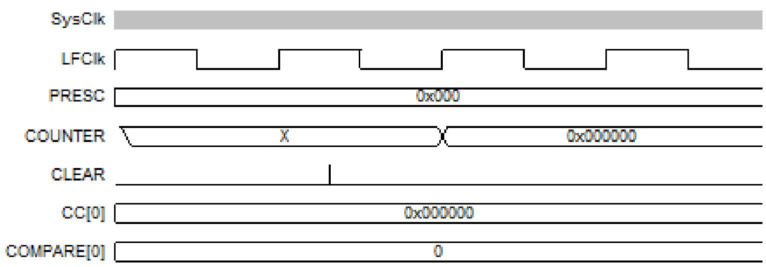



**Figure 35: Timing diagram - COMPARE_CLEAR**

- 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.

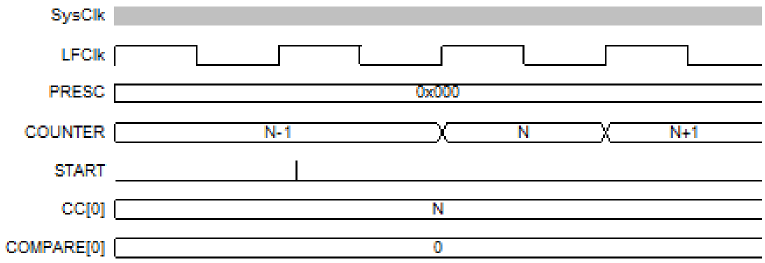



**Figure 36: Timing diagram - COMPARE_START**

- COMPARE occurs when a CC register is N and the COUNTER value transitions from N-1 to N.

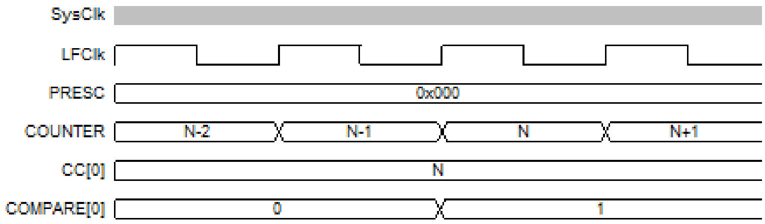



**Figure 37: Timing diagram - COMPARE**

- If the COUNTER is N, writing N+2 to a CC register is guaranteed to trigger a COMPARE event at N+2.

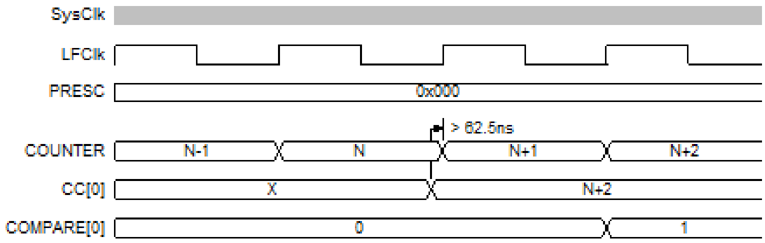



**Figure 38: Timing diagram - COMPARE_N+2**

- If the COUNTER is N, writing N or N+1 to a CC register may not trigger a COMPARE event.

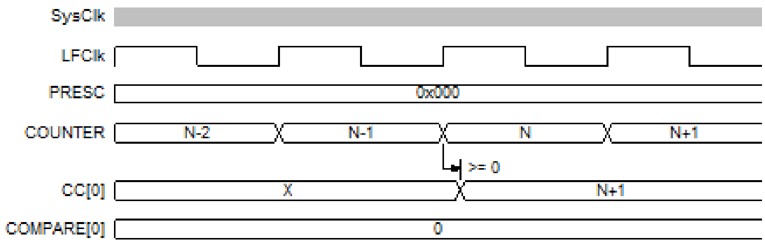



**Figure 39: Timing diagram - COMPARE_N+1**

- 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.

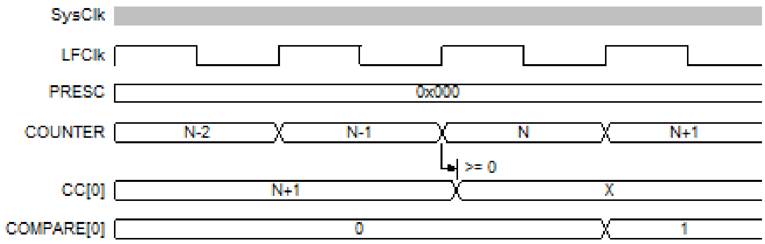



**Figure 40: Timing diagram - COMPARE_N-1**

## 19.1.8 TASK and EVENT jitter/delay

The source of 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 register called COUNTER in the LFCLK domain. COUNTER is the register which is actually modified each time the RTC ticks. These registers must be synchronised between clock domains (PCLK16M and LFCLK).

The following is a summary of the jitter introduced on tasks and events. Figures illustrating jitter follow.

### Table 150: RTC jitter magnitudes on tasks

| Task | Delay |
|---|---|
| CLEAR, STOP, START, TRIGOVRFLOW | +15 to 46 µs |

### Table 151: RTC jitter magnitudes on events

| Operation/Function | Jitter |
|---|---|
| START to COUNTER increment | +/- 15 µs |
| COMPARE to COMPARE [6] | +/- 62.5 ns |

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.

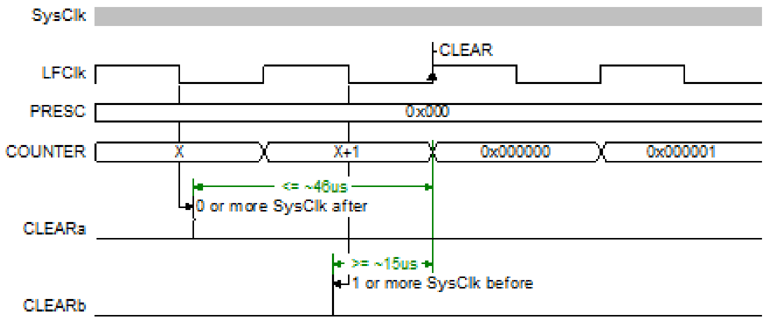



**Figure 41: Timing diagram - DELAY_CLEAR**

[6] Assumes RTC runs continuously between these events.

**Note:** 32.768 kHz clock jitter is additional to the above provided numbers.

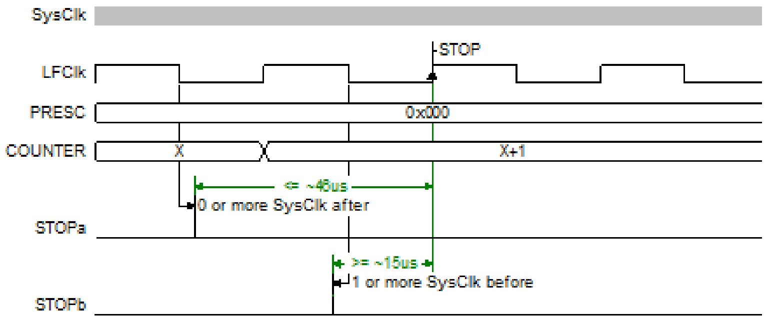



**Figure 42: Timing diagram - DELAY_STOP**

2. The START task will start the RTC. The first increment of COUNTER (and instance of TICK event) will be after 30.5 µs +/-15 µs, again because at least 1 falling edge must occur after the START TASK before the rising edge causes events and COUNTER increment. The figures show the smallest and largest delays to on the START task which appears as a +/-15 µs jitter on the first COUNTER increment.

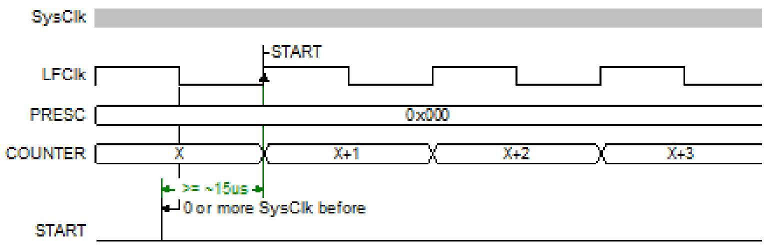



**Figure 43: Timing diagram - JITTER_START-**

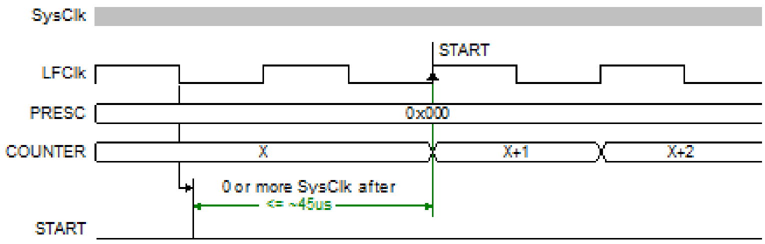



**Figure 44: Timing diagram - JITTER_START+**

## 19.1.9 Reading the COUNTER register

To read the COUNTER register, the internal <<COUNTER>> value is sampled. To ensure <<COUNTER>> is safely sampled (considering a LFCLK transition may occur during a read), the CPU and core memory bus are halted for 3 cycles by lowering the core PREADY signal. The Read takes the CPU 2 cycles in addition resulting in the COUNTER register read taking a fixed five PCLK16M clock cycles.

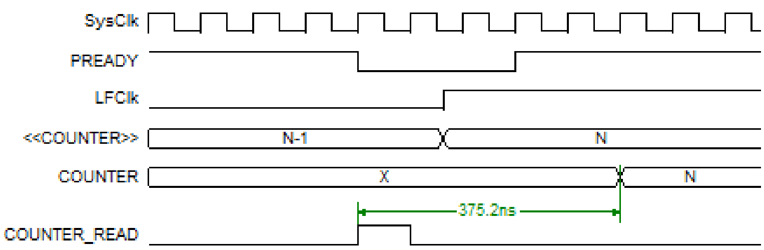



**Figure 45: Timing diagram - COUNTER_READ**

## 19.2 Register Overview

### Table 152: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x4000B000 | RTC | RTC0 | |
| 0x40011000 | RTC | RTC1 | |

### Table 153: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *START* | 0x000 | Start RTC COUNTER |
| *STOP* | 0x004 | Stop RTC COUNTER |
| *CLEAR* | 0x008 | Clear RTC COUNTER |
| *TRIGOVRFLW* | 0x00C | Set COUNTER to 0xFFFFF0 |
| Events | | |
| *TICK* | 0x100 | Event on COUNTER increment |
| *OVRFLW* | 0x104 | Event on COUNTER overflow |
| *COMPARE[0]* | 0x140 | Compare event on CC[0] match |
| *COMPARE[1]* | 0x144 | Compare event on CC[1] match |
| *COMPARE[2]* | 0x148 | Compare event on CC[2] match |
| *COMPARE[3]* | 0x14C | Compare event on CC[3] match |
| Registers | | |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *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[0]* | 0x540 | Compare register 0 |
| *CC[1]* | 0x544 | Compare register 1 |
| *CC[2]* | 0x548 | Compare register 2 |
| *CC[3]* | 0x54C | Compare register 3 |

## 19.3 Register Details

### Table 154: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TICK | | | Enable or disable interrupt on *TICK* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | OVRFLW | | | Enable or disable interrupt on *OVRFLW* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| C | RW | COMPARE0 | | | Enable or disable interrupt on *COMPARE[0]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| D | RW | COMPARE1 | | | Enable or disable interrupt on *COMPARE[1]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| E | RW | COMPARE2 | | | Enable or disable interrupt on *COMPARE[2]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| F | RW | COMPARE3 | | | Enable or disable interrupt on *COMPARE[3]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 155: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TICK | | | Write '1' to Enable interrupt on *TICK* event. |
| | | | Enabled | 1 | Enable |
| B | RW | OVRFLW | | | Write '1' to Enable interrupt on *OVRFLW* event. |

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Enabled | 1 | Enable |
| C | RW | COMPARE0 | | | Write '1' to Enable interrupt on *COMPARE[0]* event. |
| | | | Enabled | 1 | Enable |
| D | RW | COMPARE1 | | | Write '1' to Enable interrupt on *COMPARE[1]* event. |
| | | | Enabled | 1 | Enable |
| E | RW | COMPARE2 | | | Write '1' to Enable interrupt on *COMPARE[2]* event. |
| | | | Enabled | 1 | Enable |
| F | RW | COMPARE3 | | | Write '1' to Enable interrupt on *COMPARE[3]* event. |
| | | | Enabled | 1 | Enable |

## Table 156: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TICK | | | Write '1' to Clear interrupt on *TICK* event. |
| | | | Disabled | 1 | Disable |
| B | RW | OVRFLW | | | Write '1' to Clear interrupt on *OVRFLW* event. |
| | | | Disabled | 1 | Disable |
| C | RW | COMPARE0 | | | Write '1' to Clear interrupt on *COMPARE[0]* event. |
| | | | Disabled | 1 | Disable |
| D | RW | COMPARE1 | | | Write '1' to Clear interrupt on *COMPARE[1]* event. |
| | | | Disabled | 1 | Disable |
| E | RW | COMPARE2 | | | Write '1' to Clear interrupt on *COMPARE[2]* event. |
| | | | Disabled | 1 | Disable |
| F | RW | COMPARE3 | | | Write '1' to Clear interrupt on *COMPARE[3]* event. |
| | | | Disabled | 1 | Disable |

## Table 157: EVTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TICK | | | Enable or disable event routing on *TICK* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | OVRFLW | | | Enable or disable event routing on *OVRFLW* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| C | RW | COMPARE0 | | | Enable or disable event routing on *COMPARE[0]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| D | RW | COMPARE1 | | | Enable or disable event routing on *COMPARE[1]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| E | RW | COMPARE2 | | | Enable or disable event routing on *COMPARE[2]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| F | RW | COMPARE3 | | | Enable or disable event routing on *COMPARE[3]* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

## Table 158: EVTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *EVTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TICK | | | Write '1' to Enable event routing on *TICK* event. |
| | | | Enabled | 1 | Enable |
| B | RW | OVRFLW | | | Write '1' to Enable event routing on *OVRFLW* event. |
| | | | Enabled | 1 | Enable |
| C | RW | COMPARE0 | | | Write '1' to Enable event routing on *COMPARE[0]* event. |
| | | | Enabled | 1 | Enable |
| D | RW | COMPARE1 | | | Write '1' to Enable event routing on *COMPARE[1]* event. |
| | | | Enabled | 1 | Enable |
| E | RW | COMPARE2 | | | Write '1' to Enable event routing on *COMPARE[2]* event. |
| | | | Enabled | 1 | Enable |
| F | RW | COMPARE3 | | | Write '1' to Enable event routing on *COMPARE[3]* event. |
| | | | Enabled | 1 | Enable |

## Table 159: EVTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *EVTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TICK | | | Write '1' to Clear event routing on *TICK* event. |
| | | | Disabled | 1 | Disable |
| B | RW | OVRFLW | | | Write '1' to Clear event routing on *OVRFLW* event. |
| | | | Disabled | 1 | Disable |
| C | RW | COMPARE0 | | | Write '1' to Clear event routing on *COMPARE[0]* event. |
| | | | Disabled | 1 | Disable |
| D | RW | COMPARE1 | | | Write '1' to Clear event routing on *COMPARE[1]* event. |
| | | | Disabled | 1 | Disable |
| E | RW | COMPARE2 | | | Write '1' to Clear event routing on *COMPARE[2]* event. |
| | | | Disabled | 1 | Disable |
| F | RW | COMPARE3 | | | Write '1' to Clear event routing on *COMPARE[3]* event. |
| | | | Disabled | 1 | Disable |

## Table 160: 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | COUNTER | | | Counter value |

## Table 161: PRESCALER

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PRESCALER | | | Prescaler value |

## Table 162: CC[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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | | | Compare value |

# 20 Watchdog timer (WDT)

## 20.1 Functional description

The watchdog is implemented as a down-counter that generates a TIMEOUT event when it wraps over after counting down to 0. The watchdog timer is started by triggering the START task, whereupon the watchdog counter is loaded with the value specified in the CRV register. This counter is also reloaded with the value specified in the CRV register when a reload request is granted.

The watchdog's timeout period is given by:

```
timeout [s] = ( CRV + 1 ) / 32768
```

When started, the watchdog will automatically force the 32.768 kHz RC oscillator on as long as no other 32.768 kHz clock source is running and generating the 32.768 kHz system clock, see *CLOCK* chapter.

### 20.1.1 Reload criteria

The watchdog has 8 separate reload request registers which shall be used to request the watchdog to reload its counter with the value specified in the CRV register. To reload the watchdog counter, the special value 0x6E524635 needs to be written to all enabled reload registers. One or more RR registers can be individually enabled through the RREN register.

### 20.1.2 Temporarily pausing the watchdog

By default the watchdog will be active counting down the down-counter while the CPU is sleeping and when it is halted by the debugger. It is however possible to configure the watchdog to automatically pause while the CPU is sleeping as well as when it is halted by the debugger.

### 20.1.3 Watchdog reset

A TIMEOUT event will automatically lead to a watchdog reset equivalent to a system reset, see *POWER chapter* for more information about reset sources. If the watchdog is configured to generate an interrupt on the TIMEOUT event, the watchdog reset will be postponed with two 32.768 kHz clock cycles after the TIMEOUT event has been generated. Once the TIMEOUT event has been generated, the impending watchdog reset will always be effectuated.

The watchdog must be configured before it is started. After it is started, the watchdog's configuration registers, which comprises registers CRV, RREN, and CONFIG, will be blocked for further configuration.

The watchdog is reset when the device is put into System OFF mode. The watchdog is also reset when the whole system is reset, except for when the system is reset through a soft reset, see *POWER chapter* for more information about reset types.

When the device starts running again, after a reset, or waking up from OFF mode, the watchdog configuration registers will be available for configuration again.

## 20.2 Register Overview

### Table 163: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40010000 | WDT | WDT | Watchdog Timer |

### Table 164: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |

| Register | Offset | Description |
|---|---|---|
| *START* | 0x000 | Start the watchdog |
| Events | | |
| *TIMEOUT* | 0x100 | Watchdog timeout |
| Registers | | |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *RUNSTATUS* | 0x400 | Run status |
| *REQSTATUS* | 0x404 | Request status |
| *CRV* | 0x504 | Counter reload value |
| *RREN* | 0x508 | Enable register for reload request registers |
| *CONFIG* | 0x50C | Configuration register |
| *RR[0]* | 0x600 | Reload request 0 |
| *RR[1]* | 0x604 | Reload request 1 |
| *RR[2]* | 0x608 | Reload request 2 |
| *RR[3]* | 0x60C | Reload request 3 |
| *RR[4]* | 0x610 | Reload request 4 |
| *RR[5]* | 0x614 | Reload request 5 |
| *RR[6]* | 0x618 | Reload request 6 |
| *RR[7]* | 0x61C | Reload request 7 |

# 20.3 Register Details

## Table 165: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TIMEOUT | | | Write '1' to Enable interrupt on *TIMEOUT* event. |
| | | | Enabled | 1 | Enable |

## Table 166: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TIMEOUT | | | Write '1' to Clear interrupt on *TIMEOUT* event. |
| | | | Disabled | 1 | Disable |

## Table 167: RUNSTATUS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RUNSTATUS | | | Indicates whether or not the watchdog is running |
| | | | NotRunning | 0 | Watchdog not running |
| | | | Running | 1 | Watchdog is running |

## Table 168: REQSTATUS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | RR0 | | | Request status for RR[0] register |
| | | | DisabledOrRequested | 0 | RR[0] register is not enabled, or are already requesting reload |
| | | | EnabledAndUnrequested | 1 | RR[0] register is enabled, and are not yet requesting reload |
| B | R | RR1 | | | Request status for RR[1] register |
| | | | DisabledOrRequested | 0 | RR[1] register is not enabled, or are already requesting reload |
| | | | EnabledAndUnrequested | 1 | RR[1] register is enabled, and are not yet requesting reload |
| C | R | RR2 | | | Request status for RR[2] register |
| | | | DisabledOrRequested | 0 | RR[2] register is not enabled, or are already requesting reload |
| | | | EnabledAndUnrequested | 1 | RR[2] register is enabled, and are not yet requesting reload |
| D | R | RR3 | | | Request status for RR[3] register |
| | | | DisabledOrRequested | 0 | RR[3] register is not enabled, or are already requesting reload |
| | | | EnabledAndUnrequested | 1 | RR[3] register is enabled, and are not yet requesting reload |
| E | R | RR4 | | | Request status for RR[4] register |
| | | | DisabledOrRequested | 0 | RR[4] register is not enabled, or are already requesting reload |
| | | | EnabledAndUnrequested | 1 | RR[4] register is enabled, and are not yet requesting reload |
| F | R | RR5 | | | Request status for RR[5] register |
| | | | DisabledOrRequested | 0 | RR[5] register is not enabled, or are already requesting reload |
| | | | EnabledAndUnrequested | 1 | RR[5] register is enabled, and are not yet requesting reload |
| G | R | RR6 | | | Request status for RR[6] 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | DisabledOrRequested | 0 | RR[6] register is not enabled, or are already requesting reload |
| | | | EnabledAndUnrequested | 1 | RR[6] register is enabled, and are not yet requesting reload |
| H | R | RR7 | | | Request status for RR[7] register |
| | | | DisabledOrRequested | 0 | RR[7] register is not enabled, or are already requesting reload |
| | | | EnabledAndUnrequested | 1 | RR[7] register is enabled, and are not yet requesting reload |

## Table 169: CRV

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | CRV | | [0x0000000F..0xFFFFFFFF | Counter reload value in number of cycles of the 32.768 kHz clock |

## Table 170: RREN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | RR0 | | | Enable or disable RR[0] register |
| | | | Disabled | 0 | Disable RR[0] register |
| | | | Enabled | 1 | Enable RR[0] register |
| B | RW | RR1 | | | Enable or disable RR[1] register |
| | | | Disabled | 0 | Disable RR[1] register |
| | | | Enabled | 1 | Enable RR[1] register |
| C | RW | RR2 | | | Enable or disable RR[2] register |
| | | | Disabled | 0 | Disable RR[2] register |
| | | | Enabled | 1 | Enable RR[2] register |
| D | RW | RR3 | | | Enable or disable RR[3] register |
| | | | Disabled | 0 | Disable RR[3] register |
| | | | Enabled | 1 | Enable RR[3] register |
| E | RW | RR4 | | | Enable or disable RR[4] register |
| | | | Disabled | 0 | Disable RR[4] register |
| | | | Enabled | 1 | Enable RR[4] register |
| F | RW | RR5 | | | Enable or disable RR[5] register |
| | | | Disabled | 0 | Disable RR[5] register |
| | | | Enabled | 1 | Enable RR[5] register |
| G | RW | RR6 | | | Enable or disable RR[6] register |
| | | | Disabled | 0 | Disable RR[6] register |
| | | | Enabled | 1 | Enable RR[6] register |
| H | RW | RR7 | | | Enable or disable RR[7] register |
| | | | Disabled | 0 | Disable RR[7] register |
| | | | Enabled | 1 | Enable RR[7] register |

## Table 171: CONFIG

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | SLEEP | | | Configure the watchdog to either be paused, or kept running, while the CPU is sleeping |
| | | | Pause | 0 | Pause watchdog while the CPU is sleeping |
| | | | Run | 1 | Keep the watchdog running while the CPU is sleeping |
| C | RW | HALT | | | Configure the watchdog to either be paused, or kept running, while the CPU is halted by the debugger |
| | | | Pause | 0 | Pause watchdog while the CPU is halted by the debugger |
| | | | Run | 1 | Keep the watchdog running while the CPU is halted by the debugger |

## Table 172: RR[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 | A | A | A | A | A | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RR | | | Reload request register |
| | | | Reload | 0x6E524635 | Value to request a reload of the watchdog timer |

# 21 Random Number Generator (RNG)

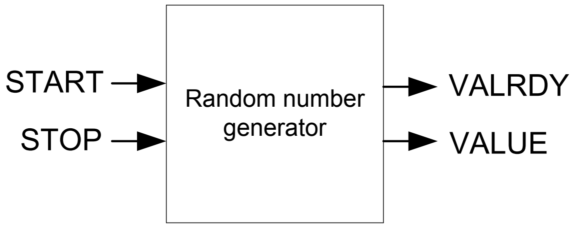



**Figure 46: Random Number Generator**

## 21.1 Functional description

The Random Number Generator (RNG) generates true non-deterministic random numbers based on internal thermal noise.

The RNG is started by triggering the START task and stopped by triggering the STOP task. When started, new random numbers are generated continuously and written to the VALUE register when ready. A VALRDY event is generated for every new random number that is written to the VALUE register. This means that after a VALRDY event is generated the CPU has the time until the next VALRDY event to read out the random number from the VALUE register before it is overwritten by a new random number.

### 21.1.1 Digital error correction

A digital corrector algorithm is employed on the internal bit stream to remove any bias toward '1' or '0'. The bits are then queued into an 8 bit register for parallel readout from the VALUE register.

It is possible to disable the bias in the CONFIG register. This offers a substantial speed advantage, but may result in a statistical distribution that is not perfectly uniform.

### 21.1.2 Speed

The time needed to generate one random byte of data is unpredictable, and may vary from one byte to the next, see product specification for more information. This is especially true when digital error correction is enabled.

## 21.2 Register Overview

**Table 173: Instances**

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x4000D000 | RNG | RNG | Random Number Generator |

**Table 174: Register Overview**

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *START* | 0x000 | Task starting the random number generator |
| *STOP* | 0x004 | Task stopping the random number generator |
| Events | | |
| *VALRDY* | 0x100 | Event being generated for every new random number written to the VALUE register |
| Registers | | |
| *SHORTS* | 0x200 | Shortcut register |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *CONFIG* | 0x504 | Configuration register |
| *VALUE* | 0x508 | Output random number |

## 21.3 Register Details

### Table 175: SHORTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | VALRDY_STOP | | | Shortcut between *VALRDY* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

### Table 176: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | VALRDY | | | Enable or disable interrupt on *VALRDY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 177: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | VALRDY | | | Write '1' to Enable interrupt on *VALRDY* event. |
| | | | Enabled | 1 | Enable |

### Table 178: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | VALRDY | | | Write '1' to Clear interrupt on *VALRDY* event. |
| | | | Disabled | 1 | Disable |

### Table 179: CONFIG

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DERCEN | | | Digital error correction |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |

### Table 180: 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | VALUE | | [0..255] | Generated random number |

# 22 Temperature sensor (TEMP)

## 22.1 Functional description

The temperature sensor measures the silicon die temperature.

The TEMP is started by triggering the START task. When the temperature measurement is completed, a DATARDY event will be generated and the result of the measurement can be read from the TEMP register

In order to be accurate, the measurement has to be performed while the HFCLK crystal oscillator is selected as clock source, see *CLOCK* for more information.

When the temperature measurement is completed, the TEMP analog electronics power down to save power.

The TEMP only supports one-shot operation, meaning that every TEMP measurement has to be explicitly started using the START task.

## 22.2 Register Overview

### Table 181: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x4000C000 | TEMP | TEMP | Temperature Sensor |

### Table 182: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *START* | 0x000 | Start temperature measurement |
| *STOP* | 0x004 | Stop temperature measurement |
| Events | | |
| *DATARDY* | 0x100 | Temperature measurement complete, data ready |
| Registers | | |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *TEMP* | 0x508 | Temperature in °C |

## 22.3 Register Details

### Table 183: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DATARDY | | | Enable or disable interrupt on *DATARDY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 184: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DATARDY | | | Write '1' to Enable interrupt on *DATARDY* event. |
| | | | Enabled | 1 | Enable |

## Table 185: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | | | | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DATARDY | | | Write '1' to Clear interrupt on *DATARDY* event. |
| | | | Disabled | 1 | Disable |

## Table 186: TEMP

| Bit number | | | | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | TEMP | | | Temperature in °C<br>Result of temperature measurement. Die temperature in °C, 2's complement format, 0.25 °C<br>Decision point: DATARDY |

# 23 AES Electronic Codebook mode encryption (ECB)

## 23.1 Functional description

AES ECB is a single AES block encrypt hardware module.

AES ECB features:

- 128 bit AES encryption
- Supports standard AES ECB block encryption
- Memory pointer support
- DMA data transfer

AES ECB performs a 128 bit AES block encrypt. At the STARTECB task, data and key is loaded into the algorithm by EasyDMA. When output data has been written back to memory, the ENDECB event is triggered.

AES ECB can be stopped by triggering the STOPECB task.

### 23.1.1 EasyDMA

The ECB implements an EasyDMA mechanism for reading and writing to the RAM. This DMA cannot access the program memory or any other parts of the memory area except RAM.

If the ECBDATAPTR is not pointing to the Data RAM region, an EasyDMA transfer may result in a HardFault or RAM corruption. See *Memory* on page 15 for more information about the different memory regions.

The EasyDMA will have finished accessing the RAM when the ENDECB or ERRORECB is generated.

### 23.1.2 ECB Data Structure

Input to the block encrypt and output from the block encrypt are stored in the same data structure. ECBDATAPTR should point to this data structure before STARTECB is initiated.

#### Table 187: ECB data structure overview

| Property | Address offset | Description |
|---|---|---|
| KEY | 0 | 16 byte AES key |
| CLEARTEXT | 16 | 16 byte AES cleartext input block |
| CIPHERTEXT | 32 | 16 byte AES ciphertext output block |

### 23.1.3 Shared resources

The ECB, CCM, and AAR share the same AES module. The ECB will always have lowest priority and if there is a sharing conflict during encryption, the ECB operation will be aborted and an ERRORECB event will be generated.

## 23.2 Register Overview

#### Table 188: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x4000E000 | ECB | ECB | AES ECB Mode Encryption |

#### Table 189: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *STARTECB* | 0x000 | Start ECB block encrypt |
| *STOPECB* | 0x004 | Abort a possible executing ECB operation |
| Events | | |
| *ENDECB* | 0x100 | ECB block encrypt complete |
| *ERRORECB* | 0x104 | ECB block encrypt aborted because of a STOPECB task or due to an error |
| Registers | | |
| *INTENSET* | 0x304 | Enable interrupt |

| Register | Offset | Description |
|---|---|---|
| *INTENCLR* | 0x308 | Disable interrupt |
| *ECBDATAPTR* | 0x504 | ECB block encrypt memory pointers |

# 23.3 Register Details

## Table 190: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENDECB | | | Write '1' to Enable interrupt on *ENDECB* event. |
| | | | Enabled | 1 | Enable |
| B | RW | ERRORECB | | | Write '1' to Enable interrupt on *ERRORECB* event. |
| | | | Enabled | 1 | Enable |

## Table 191: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENDECB | | | Write '1' to Clear interrupt on *ENDECB* event. |
| | | | Disabled | 1 | Disable |
| B | RW | ERRORECB | | | Write '1' to Clear interrupt on *ERRORECB* event. |
| | | | Disabled | 1 | Disable |

## Table 192: ECBDATAPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ECBDATAPTR | | | Pointer to the ECB data structure (see Table 1 ECB data structure overview) |

# 24 AES CCM Mode Encryption (CCM)

## 24.1 Functional description

The AES CCM supports three operations: key-stream generation, packet encryption, and packet decryption. All these operations are done in compliance with the *Bluetooth* specification[7]. A new key-stream must be generated before a new packet encryption or packet decryption operation can be started.

A key-stream is generated by triggering the KSGEN task. An ENDKSGEN event will be generated when the new key-stream has been generated. The key-stream will be stored in the AES CCM's temporary memory area, specified by the SCRATCHPTR, where it will be used in subsequent encryption and decryption operations.

Encryption is started by triggering the CRYPT task with the MODE register set to ENCRYPTION. Similarly, decryption is started by triggering the same task with MODE set to DECRYPTION. An ENDCRYPT event will be generated when packet encryption is completed as well as when packet decryption is completed, see *Figure 47: Key-stream generation followed by encryption or decryption. The shortcut is optional.* on page 121.

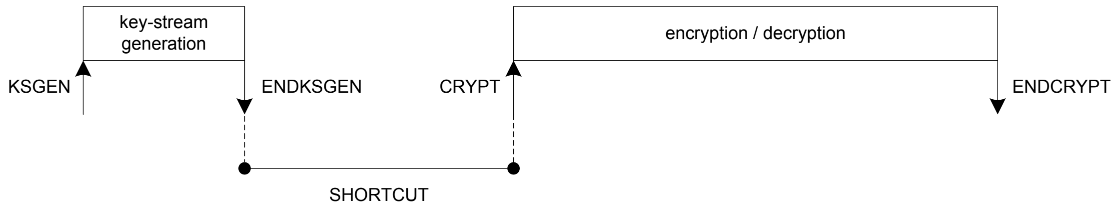



**Figure 47: Key-stream generation followed by encryption or decryption. The shortcut is optional.**

Key-stream generation, packet encryption, and packet decryption operations utilize the configuration specified in the data structure pointed to by the CNFPTR pointer. It is necessary to configure this pointer and its underlying data structure, and the MODE register before the KSGEN task is triggered. It is also necessary to configure the INPTR pointer and the OUTPTR pointer before the CRYPT task is triggered.

If a shortcut is used between ENDKSGEN event and CRYPT task, the INPTR pointer and the OUTPTR pointer must be configured before the KSGEN task is triggered.

### 24.1.1 Encryption

During packet encryption the AES CCM will read the unencrypted packet located in RAM at the address specified in the INPTR pointer, encrypt the packet and append a four byte long Message Integrity Check (MIC) field to the packet. The AES CCM will also modify the length field of the packet to adjust for the appended MIC field, that is, add four bytes to the length, and store the resulting packet back into RAM at the address specified in the OUTPTR pointer, see *Figure 48: Encryption* on page 121.

Empty packets (length field is set to 0) will not be encrypted but instead moved unmodified through the AES CCM.

The AES CCM is limited to read maximum 27 bytes of the unencrypted payload (PL) regardless of what is specified in the length field of the unencrypted packet.

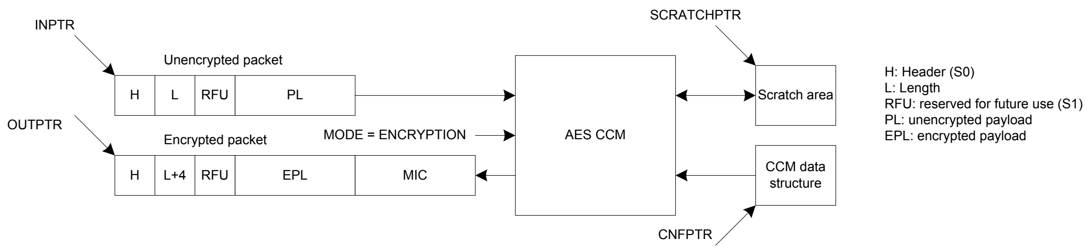



**Figure 48: Encryption**

7 *Bluetooth* AES CCM 128 bit block encryption, see *Bluetooth* Core specification Version 4.0.

## 24.1.2 Decryption

During packet decryption the AES CCM will read the encrypted packet located in RAM at the address specified in the INPTR pointer, decrypt the packet, authenticate the packet's MIC field and generate the appropriate MIC status. The AES CCM will also modify the length field of the packet to adjust for the MIC field, that is, subtract four bytes from the length, and then store the decrypted packet back into RAM at the address pointed to by the OUTPTR pointer, see *Figure 49: Decryption* on page 122.

The CCM is only able to decrypt packets that are at least 5 bytes long, that is, 1 byte encrypted payload (EPL) and 4 bytes of MIC. The CCM will therefore generate a MIC error for packets where the length field is set to 1, 2, 3 or 4.

Empty packets (length field is set to 0) will not be decrypted but instead moved unmodified through the AES CCM, these packets will always pass the MIC check.

The AES CCM is limited to read maximum 27 bytes of the encrypted payload and four bytes of the MIC regardless of what is specified in the length field of the encrypted packet.

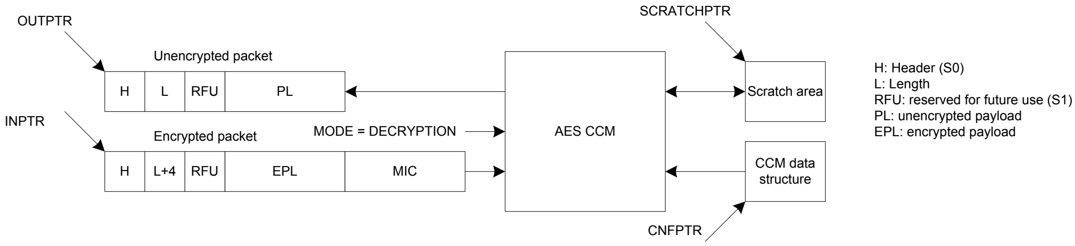



**Figure 49: Decryption**

## 24.1.3 AES CCM and RADIO concurrent operation

The AES CCM is designed to run in parallel with the RADIO to enable on-the-fly encryption and decryption of RADIO packets without CPU involvement. To facilitate this, the RADIO has to be configured with the following settings:

### Table 193: Radio configuration settings

| Radio parameter | Value | Description |
|---|---|---|
| PCNF0.S0LEN | 1 | Length of HEADER field in : *Table 195: Data structure for unencrypted packet* on page 124 and *Table 196: Data structure for encrypted packet* on page 124. |
| PCNF0.LFLEN | 5 | Length of LENGTH field in: *Table 195: Data structure for unencrypted packet* on page 124 and *Table 196: Data structure for encrypted packet* on page 124. |
| PCNF0.S1LEN | 3 | Length of the RFU field in: *Table 195: Data structure for unencrypted packet* on page 124 and *Table 196: Data structure for encrypted packet* on page 124. The combined length of LENGTH and RFU must be 8 bit always. |
| MODE | Ble_1Mbit | Data rate. |
| PCNF1.BALEN | 3 | Length of address (32 bit) |
| CRCCNF.LEN | 3 | Length of CRC (24 bit) |

## 24.1.4 Encrypting packets on-the-fly in radio transmit mode

When the AES CCM is encrypting a packet on-the-fly at the same time as the RADIO is transmitting it, the RADIO must read the encrypted packet from the same memory location as the AES CCM is writing to. The OUTPTR pointer in the AES CCM must therefore point to the same memory location as the PACKETPTR pointer in the RADIO, see *Figure 50: Configuration of on-the-fly encryption* on page 123.

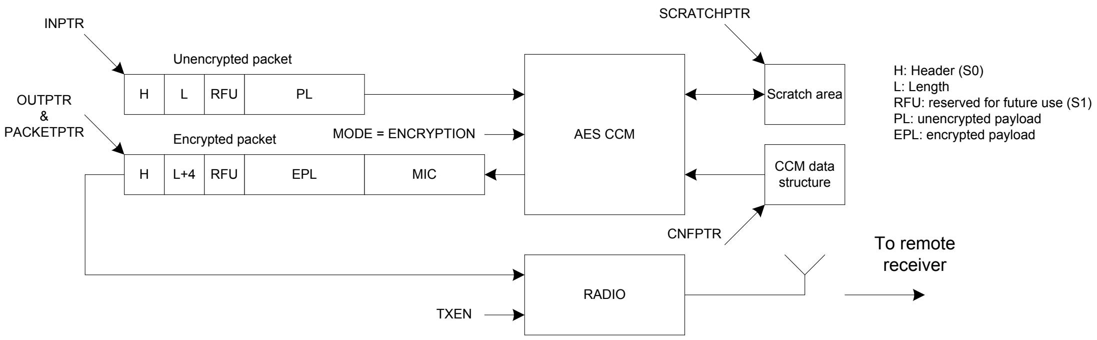



**Figure 50: Configuration of on-the-fly encryption**

In order to match the RADIO's timing, the KSGEN task must be triggered no later than when the START task in the RADIO is triggered, in addition the shortcut between the ENDKSGEN event and the CRYPT task must be enabled. This use-case is illustrated in *Figure 51: On-the-fly encryption using a PPI connection* on page 123 using a PPI connection between the READY event in the RADIO and the KSGEN task in the AES CCM.

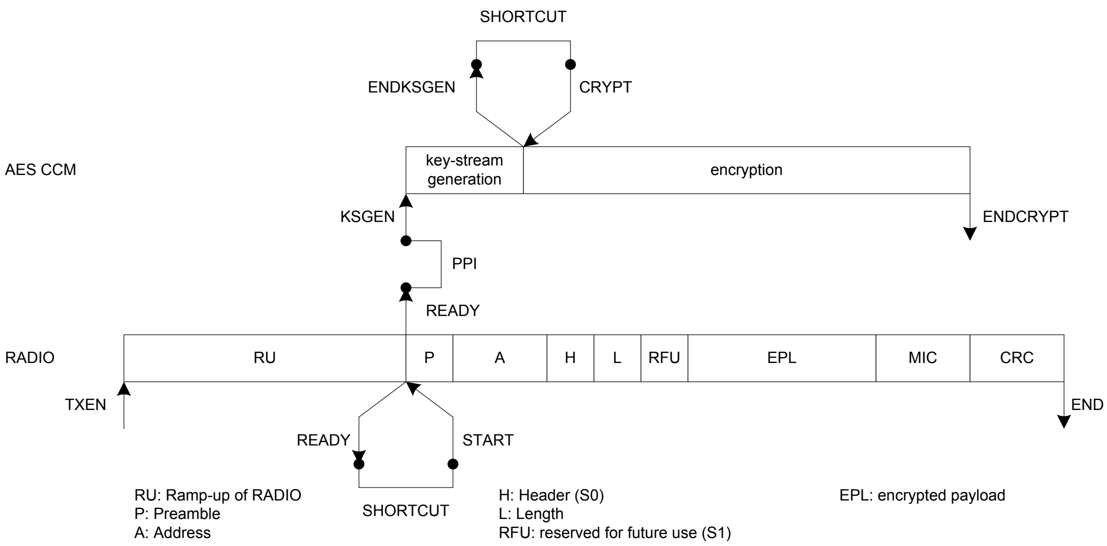



**Figure 51: On-the-fly encryption using a PPI connection**

## 24.1.5 Decrypting packets on-the-fly in radio receive mode

When the AES CCM is decrypting a packet on-the-fly at the same time as the RADIO is receiving it, the AES CCM must read the encrypted packet from the same memory location as the RADIO is writing to. The INPTR pointer in the AES CCM must therefore point to the same memory location as the PACKETPTR pointer in the RADIO, see *Figure 52: Configuration of on-the-fly decryption* on page 123.

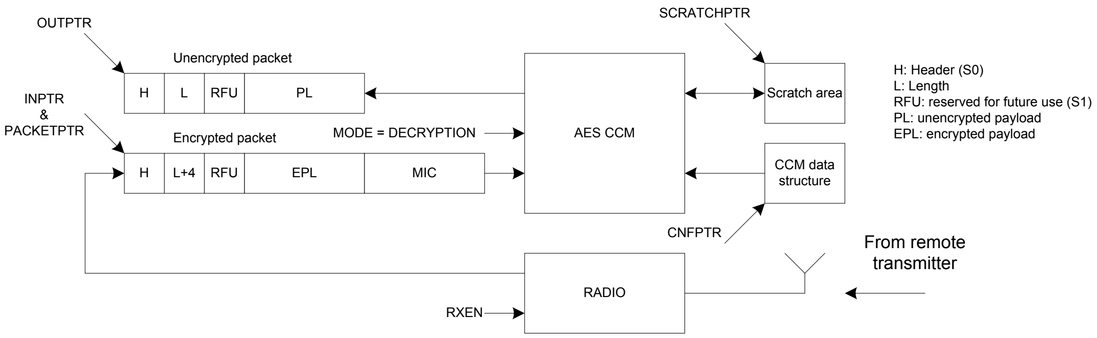



**Figure 52: Configuration of on-the-fly decryption**

In order to match the RADIO's timing, the KSGEN task must be triggered no later than when the START task in the RADIO is triggered. In addition, the CRYPT task must be triggered no earlier than when the ADDRESS event is generated by the RADIO.

If the CRYPT task is triggered exactly at the same time as the ADDRESS event is generated by the RADIO, the AES CCM will guarantee that the decryption is completed no later than when the END event in the RADIO is generated.

This use-case is illustrated in *Figure 53: On-the-fly decryption using a PPI connection between the READY event in the RADIO and the KSGEN task in the AES CCM* on page 124 using a PPI connection between the ADDRESS event in the RADIO and the CRYPT task in the AES CCM. The KSGEN task is triggered from the READY event in the RADIO through a PPI connection.

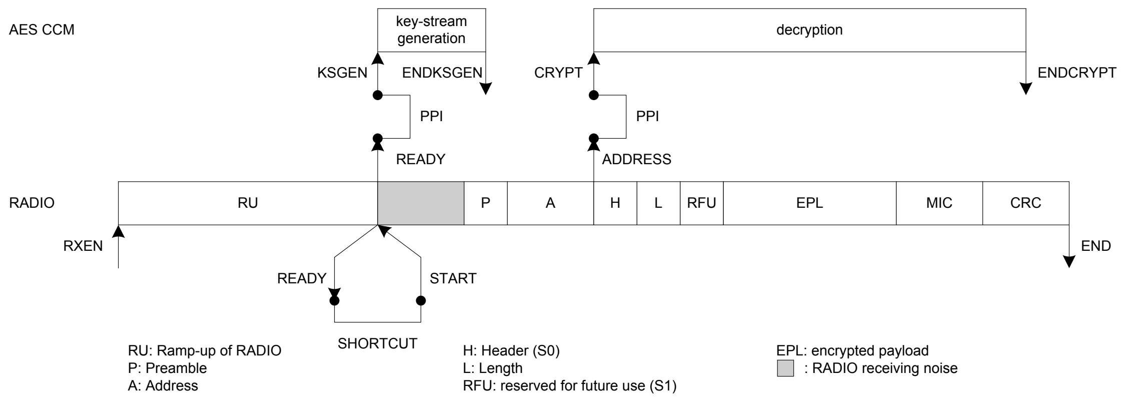



**Figure 53: On-the-fly decryption using a PPI connection between the READY event in the RADIO and the KSGEN task in the AES CCM**

## 24.1.6 CCM data structure

The CCM data structure specified in *Table 194: CCM data structure overview* on page 124 is located in RAM at the memory location specified by the CNFPTR pointer register.

### Table 194: CCM data structure overview

| Property | Address offset | Description |
|---|---|---|
| KEY | 0 | 16 byte AES key |
| PKTCTR | 16 | Octet0 (LSO) of packet counter |
| | 17 | Octet1 of packet counter |
| | 18 | Octet2 of packet counter |
| | 19 | Octet3 of packet counter |
| | 20 | Bit 6 – Bit 0: Octet4 (7 most significant bits of packet counter, with Bit 6 being the most significant bit) Bit7: Ignored |
| | 21 | Ignored |
| | 22 | Ignored |
| | 23 | Ignored |
| | 24 | Bit 0: Direction bit Bit 7 – Bit 1: Zero padded |
| IV | 25 | 8 byte initialization vector (IV) Octet0 (LSO) of IV, Octet1 of IV, … , Octet7 (MSO) of IV |

The NONCE vector (as specified by the Bluetooth Core Specification) will be generated by hardware based on the information specified in the CNFPTR data structure from *Table 194: CCM data structure overview* on page 124.

### Table 195: Data structure for unencrypted packet

| Property | Address offset | Description |
|---|---|---|
| HEADER | 0 | Packet Header |
| LENGTH | 1 | Number of bytes in unencrypted payload |
| RFU | 2 | Reserved Future Use |
| PAYLOAD | 3 | Unencrypted payload |

### Table 196: Data structure for encrypted packet

| Property | Address offset | Description |
|---|---|---|
| HEADER | 0 | Packet Header |
| LENGTH | 1 | Number of bytes in encrypted payload including length of MIC<br><br>**Note:** LENGTH will be 0 for empty packets since the MIC is not added to empty packets |
| RFU | 2 | Reserved Future Use |
| PAYLOAD | 3 | Encrypted payload |

| Property | Address offset | Description |
|---|---|---|
| MIC | 3 + payload length | ENCRYPT: 4 bytes encrypted MIC<br>**Note:** MIC is not added to empty packets |

### 24.1.7 EasyDMA and ERROR event

The CCM implements an EasyDMA mechanism for reading and writing to the RAM.

In some scenarios where the CPU and other DMA enabled peripherals are accessing the RAM at the same time, the CCM DMA could experience some bus conflicts which may also result in an error during encryption. If this happens, the ERROR event will be generated.

The EasyDMA will have finished accessing the RAM when the ENDKSGEN and ENDCRYPT events are generated.

If the CNFPTR, SCRATCHPTR, INPTR and the OUTPTR are not pointing to the Data RAM region, an EasyDMA transfer may result in a HardFault or RAM corruption. See *Memory* on page 15 for more information about the different memory regions.

### 24.1.8 Shared resources

The CCM shares registers and other resources with other peripherals that have the same ID as the CCM. The user must therefore disable all peripherals that have the same ID as the CCM before the CCM can be configured and used. Disabling a peripheral that have the same ID as the CCM will not reset any of the registers that are shared with the CCM. It is therefore important to configure all relevant CCM registers explicitly to secure that it operates correctly.

See the Instantiation table in *Instantiation* on page 17 for details on peripherals and their IDs.

## 24.2 Register Overview

### Table 197: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x4000F000 | CCM | CCM | AES CCM Mode Encryption |

### Table 198: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *KSGEN* | 0x000 | Start generation of key-stream. This operation will stop by itself when completed. |
| *CRYPT* | 0x004 | Start encryption/decryption. This operation will stop by itself when completed. |
| *STOP* | 0x008 | Stop encryption/decryption |
| Events | | |
| *ENDKSGEN* | 0x100 | Key-stream generation complete |
| *ENDCRYPT* | 0x104 | Encrypt/decrypt complete |
| *ERROR* | 0x108 | CCM error event |
| Registers | | |
| *SHORTS* | 0x200 | Shortcut register |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *MICSTATUS* | 0x400 | MIC check result |
| *ENABLE* | 0x500 | Enable |
| *MODE* | 0x504 | Operation mode |
| *CNFPTR* | 0x508 | Pointer to data structure holding AES key and NONCE vector |
| *INPTR* | 0x50C | Input pointer |
| *OUTPTR* | 0x510 | Output pointer |
| *SCRATCHPTR* | 0x514 | Pointer to data area used for temporary storage |

## 24.3 Register Details

### Table 199: SHORTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENDKSGEN_CRYPT | | | Shortcut between *ENDKSGEN* event and *CRYPT* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

## Table 200: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENDKSGEN | | | Write '1' to Enable interrupt on *ENDKSGEN* event. |
| | | | Enabled | 1 | Enable |
| B | RW | ENDCRYPT | | | Write '1' to Enable interrupt on *ENDCRYPT* event. |
| | | | Enabled | 1 | Enable |
| C | RW | ERROR | | | Write '1' to Enable interrupt on *ERROR* event. |
| | | | Enabled | 1 | Enable |

## Table 201: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENDKSGEN | | | Write '1' to Clear interrupt on *ENDKSGEN* event. |
| | | | Disabled | 1 | Disable |
| B | RW | ENDCRYPT | | | Write '1' to Clear interrupt on *ENDCRYPT* event. |
| | | | Disabled | 1 | Disable |
| C | RW | ERROR | | | Write '1' to Clear interrupt on *ERROR* event. |
| | | | Disabled | 1 | Disable |

## Table 202: MICSTATUS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | MICSTATUS | | | The result of the MIC check performed during the previous decryption operation |
| | | | CheckFailed | 0 | MIC check failed |
| | | | CheckPassed | 1 | MIC check passed |

## Table 203: 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 | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | Enable or disable CCM |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 2 | Enable |

## Table 204: 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | MODE | | | The mode of operation to be used |
| | | | Encryption | 0 | AES CCM packet encryption mode |
| | | | Decryption | 1 | AES CCM packet decryption mode |

## Table 205: CNFPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CNFPTR | | | Pointer to the data structure holding the AES key and the CCM NONCE vector (see Table 1 CCM data structure overview) |

## Table 206: INPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | INPTR | | | Input pointer |

## Table 207: OUTPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OUTPTR | | | Output pointer |

## Table 208: SCRATCHPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | SCRATCHPTR | | | Pointer to a "scratch" data area used for temporary storage during key-stream generation, MIC generation and encryption/ decryption.The scratch area is used for temporary storage of data during key-stream generation and encryption. 16 + MAXPACKETSIZE number of bytes must be reserved in RAM for this area. |

# 25 Accelerated Address Resolver (AAR)

## 25.1 Functional description

### 25.1.1 Resolving a resolvable address

A private resolvable address shall be composed of 6 bytes as illustrated in *Figure 54: Resolvable address* on page 128

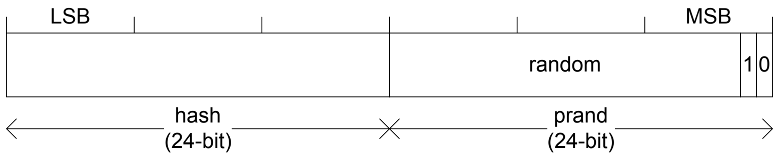



**Figure 54: Resolvable address**

To resolve an address the ADDRPTR pointer must point to the least significant byte (LSB) of the resolvable address offset by 3 bytes to accommodate the packet header. The resolver is started by triggering the START task. A RESOLVED event is generated when and if the AAR manages to resolve the address using one of the Identity Resolving Keys (IRK) found in the IRK data structure. The AAR will use the IRK specified in the register IRK0 to IRK15 starting from IRK0. How many to be used is specified by the NIRK register. The AAR module will generate a NOTRESOLVED event if it is not able to resolve the address using the specified list of IRKs.

The AAR will go through the list of available IRKs in the IRK data structure and for each IRK try to resolve the address according to the Resolvable Private Address Resolution Procedure described in the *Bluetooth* Specification[8]. The time it takes to resolve an address may vary depending on where in the list the resolvable address is located. The resolution time will also be affected by RAM accesses performed by other peripherals and the CPU. See product specification for more information about resolution time.

The AAR will not distinguish between public and random addresses. The AAR will also not distinguish between static and private addresses, or between private resolvable and private non-resolvable addresses.

The AAR will stop as soon as it has managed to resolve the address, or after trying to resolve the address using NIRK number of IRKs from the IRK data structure. The AAR will generate an END event after it has stopped.

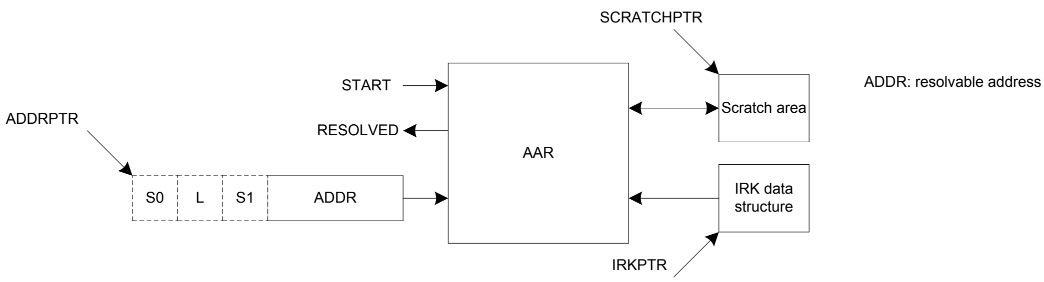



**Figure 55: Address resolution with packet preloaded into RAM**

### 25.1.2 Use case example for chaining RADIO packet reception with resolving addresses with the AAR

The AAR may be started as soon as the 6 bytes required by the AAR has been received by the RADIO and stored in RAM. The ADDRPTR pointer must point to the least significant byte of the resolvable address within the received packet offset by 3 bytes to accommodate the packet header.

8 *Bluetooth* Specification Version 4.0 [Vol 3] chapter 10.8.2.3.

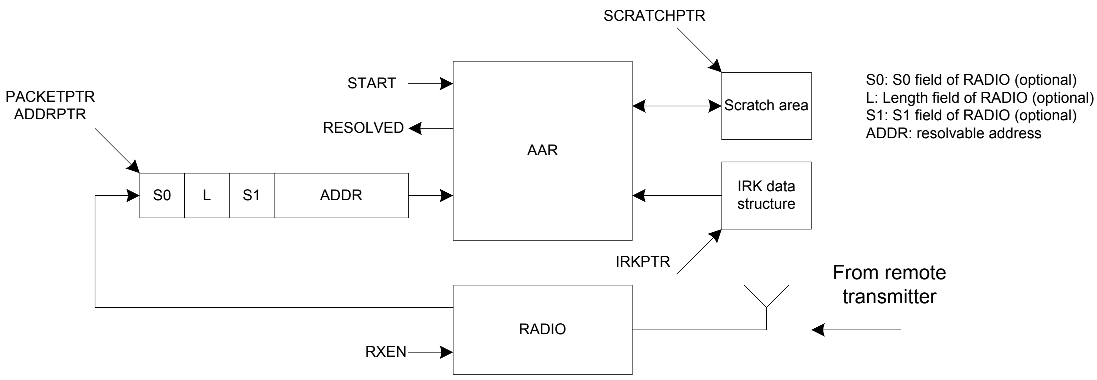



**Figure 56: Address resolution with packet loaded into RAM by the RADIO**

### 25.1.3 IRK data structure

The IRK data structure specified in *Table 209: IRK data structure overview* on page 129 is located in RAM at the memory location specified by the CNFPTR pointer register.

**Table 209: IRK data structure overview**

| Property | Address offset | Description |
|---|---|---|
| IRK0 | 0 | IRK number 0 (16 - byte) |
| IRK1 | 16 | IRK number 0 (16 - byte) |
| .. | .. | .. |
| IRK15 | 240 | IRK number 15 (16 - byte) |

### 25.1.4 EasyDMA

The AAR implements EasyDMA for reading and writing to the RAM. The EasyDMA will have finished accessing the RAM when the END, RESOLVED, and NOTRESOLVED events are generated.

If the IRKPTR, ADDRPTR and the SCRATCHPTR is not pointing to the Data RAM region, an EasyDMA transfer may result in a HardFault or RAM corruption. See *Memory* on page 15 for more information about the different memory regions.

### 25.1.5 Shared resources

The AAR shares registers and other resources with other peripherals that have the same ID as the AAR. The user must therefore disable all peripherals that have the same ID as the AAR before the AAR can be configured and used. Disabling a peripheral that have the same ID as the AAR will not reset any of the registers that are shared with the AAR. It is therefore important to configure all relevant AAR registers explicitly to secure that it operates correctly.

See the Instantiation table in *Instantiation* on page 17 for details on peripherals and their IDs.

### 25.1.6 Register Overview

**Table 210: Instances**

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x4000F000 | AAR | AAR | Accelerated Address Resolver |

**Table 211: Register Overview**

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *START* | 0x000 | Start resolving addresses based on IRKs specified in the IRK data structure |
| *STOP* | 0x008 | Stop resolving addresses |
| Events | | |
| *END* | 0x100 | Address resolution procedure complete |
| *RESOLVED* | 0x104 | Address resolved |
| *NOTRESOLVED* | 0x108 | Address not resolved |
| Registers | | |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *STATUS* | 0x400 | Resolution status |
| *ENABLE* | 0x500 | Enable AAR |
| *NIRK* | 0x504 | Number of IRKs |

| Register | Offset | Description |
|---|---|---|
| *IRKPTR* | 0x508 | Pointer to IRK data structure |
| *ADDRPTR* | 0x510 | Pointer to the resolvable address |
| *SCRATCHPTR* | 0x514 | Pointer to data area used for temporary storage |

## 25.1.7 Register Details

### Table 212: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END | | | Write '1' to Enable interrupt on *END* event. |
| | | | Enabled | 1 | Enable |
| B | RW | RESOLVED | | | Write '1' to Enable interrupt on *RESOLVED* event. |
| | | | Enabled | 1 | Enable |
| C | RW | NOTRESOLVED | | | Write '1' to Enable interrupt on *NOTRESOLVED* event. |
| | | | Enabled | 1 | Enable |

### Table 213: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END | | | Write '1' to Clear interrupt on *END* event. |
| | | | Disabled | 1 | Disable |
| B | RW | RESOLVED | | | Write '1' to Clear interrupt on *RESOLVED* event. |
| | | | Disabled | 1 | Disable |
| C | RW | NOTRESOLVED | | | Write '1' to Clear interrupt on *NOTRESOLVED* event. |
| | | | Disabled | 1 | Disable |

### Table 214: 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 | A | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | STATUS | | [0..15] | The IRK that was used last time an address was resolved |

### Table 215: 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 | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | Enable or disable AAR |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 3 | Enable |

### Table 216: NIRK

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | NIRK | | [1..16] | Number of Identity root keys available in the IRK data structure |

### Table 217: IRKPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | IRKPTR | | | Pointer to the IRK data structure |

### Table 218: ADDRPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ADDRPTR | | | Pointer to the resolvable address (6-bytes) |

**Table 219: SCRATCHPTR**

| Bit number | | | | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | SCRATCHPTR | | | Pointer to a "scratch" data area used for temporary storage during resolution.A space of minimum 3 bytes must be reserved. |

# 26 Serial Peripheral Interface (SPI) Master

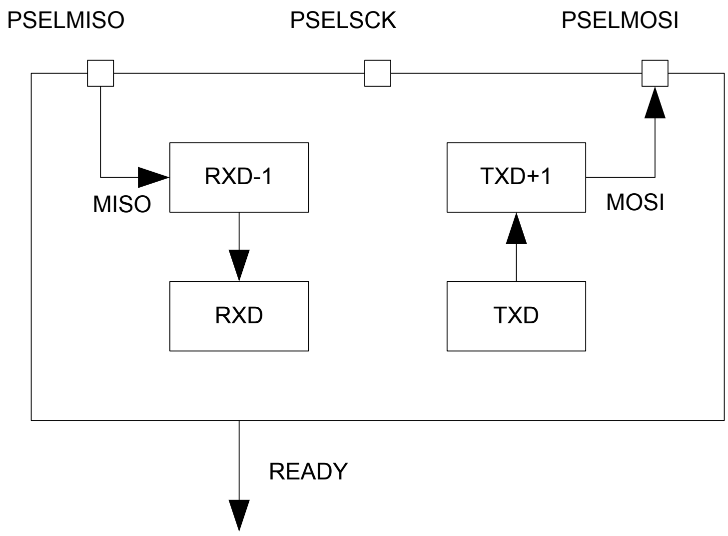



**Figure 57: SPI master**

**Note:** RXD-1 and TXD+1 illustrate the double buffered version of RXD and TXD respectively.

## 26.1 Functional description

The SPI master as illustrated in *Figure 57: SPI master* on page 132 provides a simple CPU interface which includes a TXD register for sending data and an RXD register for receiving data. These registers are double buffered to enable some degree of uninterrupted data flow in and out of the SPI master. The SPI master does not implement support for chip select directly. Therefore, the CPU must use available GPIOs to select the correct slave and control this independently of the SPI master. The SPI master supports SPI modes 0 through 3.

**Table 220: SPI modes**

| Mode | Clock polarity CPOL | Clock phase CPHA |
|---|---|---|
| SPI_MODE | 0 | 0 |
| SPI_MODE | 0 | 1 |
| SPI_MODE | 1 | 0 |
| SPI_MODE | 1 | 1 |

### 26.1.1 SPI master mode pin configuration

The different signals SCK, MOSI, and MISO associated with the SPI master are mapped to physical pins according to the configuration specified in the PSELSCK, PSELMOSI, and PSELMISO registers respectively. If a value of 0xFFFFFFFF is specified in any of these registers, the associated SPI master signal is not connected to any physical pin. The PSELSCK, PSELMOSI, and PSELMISO registers and their configurations are only used as long as the SPI master is enabled, and retained only as long as the device is in ON mode. PSELSCK, PSELMOSI, and PSELMISO must only be configured when the SPI master is disabled.

To secure correct behavior in the SPI, the pins used by the SPI must be configured in the GPIO peripheral as described in *Table 221: GPIO configuration* on page 133 prior to enabling the SPI. The SCK must always be connected to a pin, and that pin's input buffer must always be connected for the SPI to work. This configuration must be retained in the GPIO for the selected IOs as long as the SPI is enabled.

Only one peripheral can be assigned to drive a particular GPIO pin at a time, failing to do so may result in unpredictable behavior.

**Table 221: GPIO configuration**

| SPI master signal | SPI master pin | Direction | Output value |
|---|---|---|---|
| SCK | As specified in PSELSCK | Output | Same as CONFIG.CPOL |
| MOSI | As specified in PSELMOSI | Output | 0 |
| MISO | As specified in PSELMISO | Input | Not applicable |

## 26.1.2 Shared resources

The SPI shares registers and other resources with other peripherals that have the same ID as the SPI. Therefore, the user must disable all peripherals that have the same ID as the SPI before the SPI can be configured and used. Disabling a peripheral that have the same ID as the SPI will not reset any of the registers that are shared with the SPI. It is therefore important to configure all relevant SPI registers explicitly to secure that it operates correctly.

See the Instantiation table in *Instantiation* on page 17 for details on peripherals and their IDs.

## 26.1.3 SPI master transaction sequence

An SPI master transaction is started by writing the first byte, which is to be transmitted by the SPI master, to the TXD register. Since the transmitter is double buffered, the second byte can be written to the TXD register immediately after the first one. The SPI master will then send these bytes in the order they are written to the TXD register.

The SPI master is a synchronous interface, and for every byte that is sent, a different byte will be received at the same time; this is illustrated in *Figure 58: SPI master transaction* on page 133. Bytes that are received will be moved to the RXD register where the CPU can extract them by reading the register. The RXD register is double buffered in the same way as the TXD register, and a second byte can therefore be received at the same time as the first byte is being extracted from RXD by the CPU. The SPI master will generate a READY event every time a new byte is moved to the RXD register. The double buffered byte will be moved from RXD-1 to RXD as soon as the first byte is extracted from RXD. The SPI master will stop when there are no more bytes to send in TXD and TXD+1.

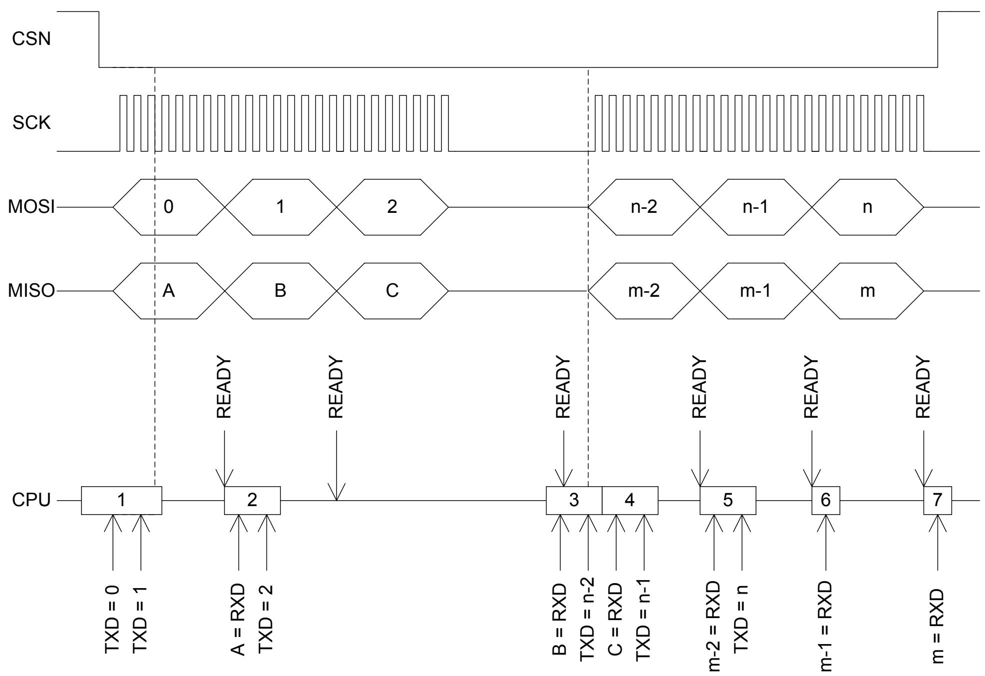



**Figure 58: SPI master transaction**

The READY event of the third byte transaction is delayed until B is extracted from RXD in occurrence number 3 on the horizontal lifeline. The reason for this is that the third event is generated first when C is moved from RXD-1 to RXD after B is read.

The SPI master will move the incoming byte to the RXD register after a short delay following the SCK clock period of the last bit in the byte. This also means that the READY event will be delayed accordingly, see *Figure 59: SPI master transaction* on page 134. Therefore, it is important that you always clear the READY event, even if the RXD register and the data that is being received is not used.

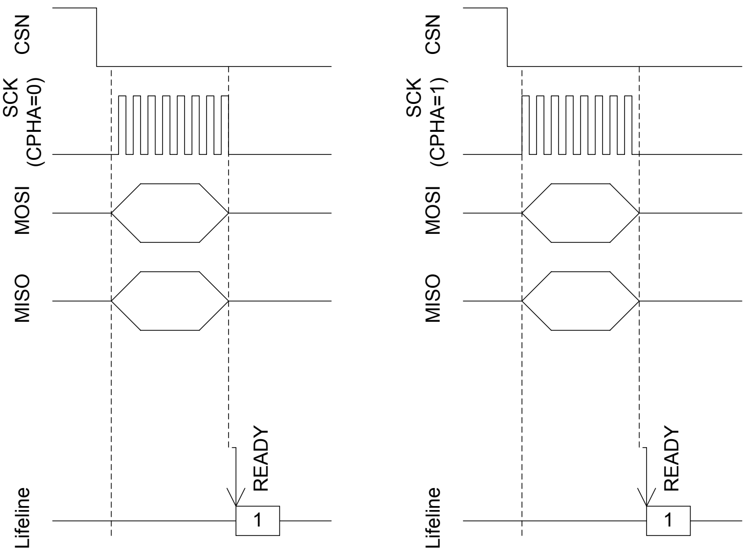



**Figure 59: SPI master transaction**

## 26.2 Register Overview

### Table 222: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40003000 | SPI | SPI0 | Serial Peripheral Interface |
| 0x40004000 | SPI | SPI1 | Serial Peripheral Interface |

### Table 223: Register Overview

| Register | Offset | Description |
|---|---|---|
| Events | | |
| *READY* | 0x108 | TXD byte sent and RXD byte received |
| Registers | | |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *ENABLE* | 0x500 | Enable SPI |
| *PSELSCK* | 0x508 | Pin select for SCK |
| *PSELMOSI* | 0x50C | Pin select for MOSI |
| *PSELMISO* | 0x510 | Pin select for MISO |
| *RXD* | 0x518 | RXD register |
| *TXD* | 0x51C | TXD register |
| *FREQUENCY* | 0x524 | SPI frequency |
| *CONFIG* | 0x554 | Configuration register |

## 26.3 Register Details

### Table 224: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY | | | Enable or disable interrupt on *READY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

## Table 225: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY | | | Write '1' to Enable interrupt on *READY* event. |
| | | | Enabled | 1 | Enable |

## Table 226: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY | | | Write '1' to Clear interrupt on *READY* event. |
| | | | Disabled | 1 | Disable |

## Table 227: 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 | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | Enable or disable SPI |
| | | | Disabled | 0 | Disable SPI |
| | | | Enabled | 1 | Enable SPI |

## Table 228: PSELSCK

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELSCK | | [0..31] | Pin number configuration for SPI SCK signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 229: PSELMOSI

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELMOSI | | [0..31] | Pin number configuration for SPI MOSI signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 230: PSELMISO

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELMISO | | [0..31] | Pin number configuration for SPI MISO signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 231: RXD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RXD | | | RX data received. Double buffered |

## Table 232: TXD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TXD | | | TX data to send. Double buffered |

## Table 233: 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 | | | | | 0 | 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 | 0 | 0 | 0 | 0 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | FREQUENCY | | | SPI master data rate |
| | | | K125 | 0x02000000 | 125 kbps |
| | | | K250 | 0x04000000 | 250 kbps |
| | | | K500 | 0x08000000 | 500 kbps |
| | | | M1 | 0x10000000 | 1 Mbps |
| | | | M2 | 0x20000000 | 2 Mbps |
| | | | M4 | 0x40000000 | 4 Mbps |
| | | | M8 | 0x80000000 | 8 Mbps |

## Table 234: CONFIG

| Bit number | | | | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ORDER | | | Bit order |
| | | | MsbFirst | 0 | Most significant bit shifted out first |
| | | | LsbFirst | 1 | Least significant bit shifted out first |
| B | RW | CPHA | | | Serial clock (SCK) phase |
| | | | Leading | 0 | Sample on leading edge of clock, shift serial data on trailing edge |
| | | | Trailing | 1 | Sample on trailing edge of clock, shift serial data on leading edge |
| C | RW | CPOL | | | Serial clock (SCK) polarity |
| | | | ActiveHigh | 0 | Active high |
| | | | ActiveLow | 1 | Active low |

# 27 SPI Slave (SPIS)

SPIS is a SPI slave with EasyDMA support for ultra low power serial communication from an external SPI master. EasyDMA in conjunction with hardware based semaphore mechanisms removes all real-time requirements associated with controlling the SPI slave from a low priority CPU execution context.

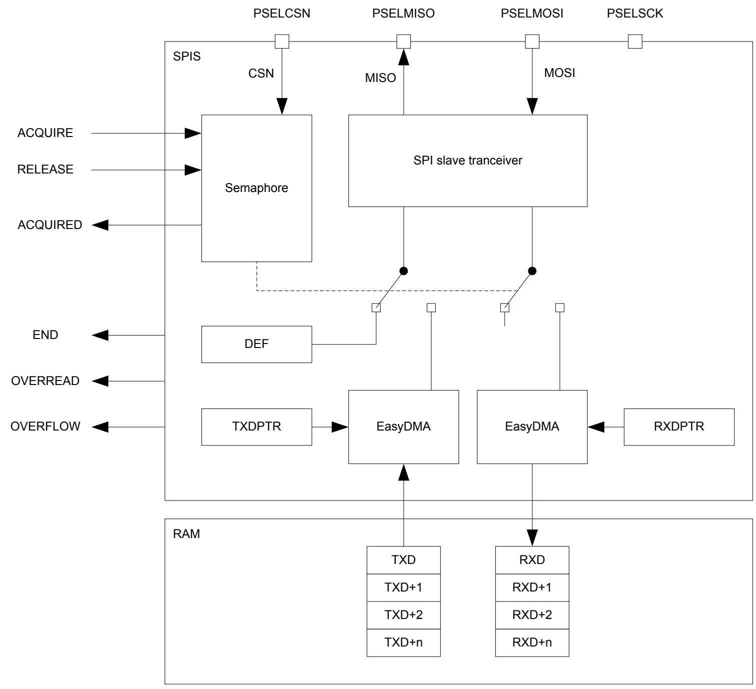



**Figure 60: SPI slave**

## 27.1 Pin configuration

The different signals CSN, SCK, MOSI, and MISO associated with the SPI slave are mapped to physical pins according to the configuration specified in the PSELCSN, PSELSCK, PSELMOSI, and PSELMISO registers respectively. If a value of 0xFFFFFFFF is specified in any of these registers, the associated SPI slave signal will not be connected to any physical pins.

The PSELCSN, PSELSCK, PSELMOSI, and PSELMISO registers and their configurations are only used as long as the SPI slave is enabled, and retained only as long as the device is in System ON mode, see *POWER* chapter for more information about power modes. PSELCSN, PSELSCK, PSELMOSI, and PSELMISO must only be configured when the SPI slave is disabled.

To secure correct behavior in the SPI slave, the pins used by the SPI slave must be configured in the GPIO peripheral as described in *Table 235: Pin configuration* on page 137 prior to enabling the SPI slave. This is to secure that the pins used by the SPI slave are driven correctly if the SPI slave 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 SPI slave is to be recognized by an external SPI master.

The MISO line is set in high impedance as long as the SPI slave is not selected with CSN.

Only one peripheral can be assigned to drive a particular GPIO pin at a time, failing to do so may result in unpredictable behavior.

**Table 235: Pin configuration**

| SPI signal | SPI pin | Direction | Output value | Comment |
|---|---|---|---|---|
| CSN | As specified in PSELCSN | Input | Not applicable | |
| SCK | As specified in PSELSCK | Input | Not applicable | |
| MOSI | As specified in PSELMOSI | Input | Not applicable | |

| SPI signal | SPI pin | Direction | Output value | Comment |
|---|---|---|---|---|
| MISO | As specified in PSELMISO | Input | Not applicable | Emulates that the SPI slave is not selected. |

## 27.2 Shared resources

The SPI slave shares registers and other resources with other peripherals that have the same ID as the SPI slave. Therefore, you must disable all peripherals that have the same ID as the SPI slave before the SPI slave can be configured and used. Disabling a peripheral that has the same ID as the SPI slave will not reset any of the registers that are shared with the SPI slave. It is important to configure all relevant SPI slave registers explicitly to secure that it operates correctly.

The Instantiation table in *Instantiation* on page 17 shows which peripherals have the same ID as the SPI slave.

## 27.3 EasyDMA

The SPI Slave implements EasyDMA for reading and writing to and from the RAM. The END event indicates that EasyDMA has finished accessing the buffer in RAM.

If the TXDPTR and the RXDPTR are not pointing to the Data RAM region, an EasyDMA transfer may result in a HardFault or RAM corruption. See *Memory* on page 15 for more information about the different memory regions.

## 27.4 SPI slave operation

SPI slave uses two memory pointers, RXDPTR and TXDPTR, that point to the RXD buffer (receive buffer) and TXD buffer (transmit buffer) respectively, see *Figure 60: SPI slave* on page 137. Since these buffers are located in RAM, which can be accessed by both the SPI slave and the CPU, a hardware based semaphore mechanism is implemented to enable safe sharing.

Before the CPU can safely update the RXDPTR and TXDPTR pointers it must first acquire the SPI semaphore. The CPU can acquire the semaphore by triggering the ACQUIRE task and then receiving the ACQUIRED event. When the CPU has updated the RXDPTR and TXDPTR pointers the CPU must release the semaphore before the SPI slave will be able to acquire it. The CPU releases the semaphore by triggering the RELEASE task. This is illustrated in *Figure 61: SPI transaction when shortcut between END and ACQUIRE is enabled* on page 139. Triggering the RELEASE task when the semaphore is not granted to the CPU will have no effect.

The semaphore mechanism does not, at any time, prevent the CPU from performing read or write access to the RXDPTR register, the TXDPTR registers, or the RAM that these pointers are pointing to. The semaphore is only telling when these can be updated by the CPU so that safe sharing is achieved.

The semaphore is by default assigned to the CPU after the SPI slave is enabled. No ACQUIRED event will be generated for this initial semaphore handover. An ACQUIRED event will be generated immediately if the ACQUIRE task is triggered while the semaphore is assigned to the CPU.

The SPI slave will try to acquire the semaphore when CSN goes low. If the SPI slave does not manage to acquire the semaphore at this point, the transaction will be ignored. This means that all incoming data on MOSI will be discarded, and the DEF (default) character will be clocked out on the MISO line throughout the whole transaction. This will also be the case even if the semaphore is released by the CPU during the transaction. In case of a race condition where the CPU and the SPI slave try to acquire the semaphore at the same time, as illustrated in lifeline item 2 in *Figure 61: SPI transaction when shortcut between END and ACQUIRE is enabled* on page 139, the semaphore will be granted to the CPU.

If the SPI slave acquires the semaphore, the transaction will be granted. The incoming data on MOSI will be stored in the RXD buffer and the data in the TXD buffer will be clocked out on MISO.

When a granted transaction is completed and CSN goes high, the SPI slave will automatically release the semaphore and generate the END event.

As long as the semaphore is available the SPI slave can be granted multiple transactions one after the other. If the CPU is not able to reconfigure the TXDPTR and RXDPTR between granted transactions, the same TX data will be clocked out and the RX buffers will be overwritten. To prevent this from happening, the END_ACQUIRE shortcut can be used. With this shortcut enabled the semaphore will be handed over to the CPU automatically after the granted transaction has completed, giving the CPU the ability to update the TXPTR and RXPTR between every granted transaction.

If the CPU tries to acquire the semaphore while it is assigned to the SPI slave, an immediate handover will not be granted. However, the semaphore will be handed over to the CPU as soon as the SPI slave has released the semaphore after the granted transaction is completed. If the END_ACQUIRE shortcut is enabled and the CPU has triggered the ACQUIRE task during a granted transaction, only one ACQUIRE request will be served following the END event.

The MAXRX register specifies the maximum number of bytes the SPI slave can receive in one granted transaction. If the SPI slave receives more than MAXRX number of bytes, an OVERFLOW will be indicated in the STATUS register and the incoming bytes will be discarded.

The MAXTX parameter specifies the maximum number of bytes the SPI slave can transmit in one granted transaction. If the SPI slave is forced to transmit more than MAXTX number of bytes, an OVERREAD will be indicated in the STATUS register and the ORC character will be clocked out.

The AMOUNTRX and AMOUNTTX registers are updated when a granted transaction is completed. The AMOUNTTX register indicates how many bytes were read from the TX buffer in the last transaction, that is, ORC (over-read) characters are not included in this number. Similarly, the AMOUNTRX register indicates how many bytes were written into the RX buffer in the last transaction.

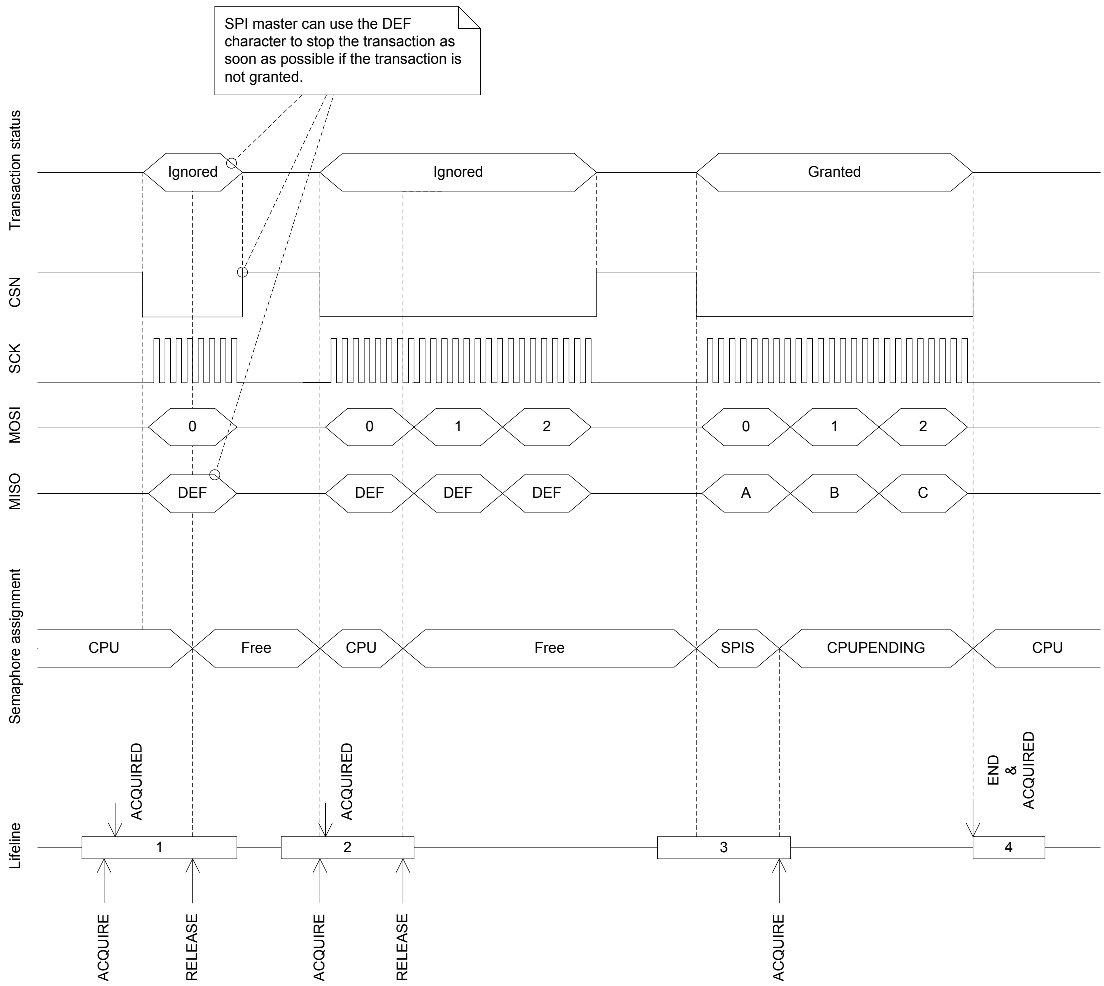



**Figure 61: SPI transaction when shortcut between END and ACQUIRE is enabled**

## 27.5 Register Overview

### Table 236: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40004000 | SPIS | SPIS1 | SPI Slave |

### Table 237: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *ACQUIRE* | 0x024 | Acquire SPI semaphore |
| *RELEASE* | 0x028 | Release SPI semaphore, enabling the SPI slave to acquire it |
| Events | | |
| *END* | 0x104 | Granted transaction completed |
| *ACQUIRED* | 0x128 | Semaphore acquired |
| Registers | | |
| *SHORTS* | 0x200 | Shortcut register |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *SEMSTAT* | 0x400 | Semaphore status register |
| *STATUS* | 0x440 | Status from last transaction |
| *ENABLE* | 0x500 | Enable SPI slave |
| *PSELSCK* | 0x508 | Pin select for SCK |
| *PSELMISO* | 0x50C | Pin select for MISO |
| *PSELMOSI* | 0x510 | Pin select for MOSI |
| *PSELCSN* | 0x514 | Pin select for CSN |
| *RXDPTR* | 0x534 | RXD data pointer |
| *MAXRX* | 0x538 | Maximum number of bytes in receive buffer |
| *AMOUNTRX* | 0x53C | Number of bytes received in last granted transaction |
| *TXDPTR* | 0x544 | TXD data pointer |
| *MAXTX* | 0x548 | Maximum number of bytes in transmit buffer |
| *AMOUNTTX* | 0x54C | Number of bytes transmitted in last granted transaction |
| *CONFIG* | 0x554 | Configuration register |
| *DEF* | 0x55C | Default character. Character clocked out in case of an ignored transaction. |
| *ORC* | 0x5C0 | Over-read character |

## 27.6 Register Details

### Table 238: SHORTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END_ACQUIRE | | | Shortcut between *END* event and *ACQUIRE* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

### Table 239: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END | | | Enable or disable interrupt on *END* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | ACQUIRED | | | Enable or disable interrupt on *ACQUIRED* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 240: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END | | | Write '1' to Enable interrupt on *END* event. |
| | | | Enabled | 1 | Enable |
| B | RW | ACQUIRED | | | Write '1' to Enable interrupt on *ACQUIRED* event. |
| | | | Enabled | 1 | Enable |

### Table 241: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END | | | Write '1' to Clear interrupt on *END* event. |
| | | | Disabled | 1 | Disable |

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| B | RW | ACQUIRED | | | Write '1' to Clear interrupt on *ACQUIRED* event. |
| | | | Disabled | 1 | Disable |

## Table 242: SEMSTAT

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | R | SEMSTAT | | | Semaphore status |
| | | | Free | 0 | Semaphore is free |
| | | | CPU | 1 | Semaphore is assigned to CPU |
| | | | SPIS | 2 | Semaphore is assigned to SPI slave |
| | | | CPUPending | 3 | Semaphore is assigned to SPI but a handover to the CPU is pending |

## Table 243: STATUS

**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OVERREAD | | | TX buffer over-read detected, and prevented |
| | | | NotPresent | 0 | Read: error not present |
| | | | Present | 1 | Read: error present |
| | | | Clear | 1 | Write: clear error on writing '1' |
| B | RW | OVERFLOW | | | RX buffer overflow detected, and prevented |
| | | | NotPresent | 0 | Read: error not present |
| | | | Present | 1 | Read: error present |
| | | | Clear | 1 | Write: clear error on writing '1' |

## Table 244: 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 | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | Enable or disable SPI slave |
| | | | Disabled | 0 | Disable SPI slave |
| | | | Enabled | 2 | Enable SPI slave |

## Table 245: PSELSCK

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELSCK | | [0..31] | Pin number configuration for SPI SCK signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 246: PSELMISO

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELMISO | | [0..31] | Pin number configuration for SPI MISO signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 247: PSELMOSI

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELMOSI | | [0..31] | Pin number configuration for SPI MOSI signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

### Table 248: PSELCSN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELCSN | | [0..31] | Pin number configuration for SPI CSN signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

### Table 249: RXDPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | RXDPTR | | | RXD data pointer |

### Table 250: MAXRX

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | MAXRX | | | Maximum number of bytes in receive buffer |

### Table 251: AMOUNTRX

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | AMOUNTRX | | | Number of bytes received in the last granted transaction |

### Table 252: TXDPTR

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TXDPTR | | | TXD data pointer |

### Table 253: MAXTX

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | MAXTX | | | Maximum number of bytes in transmit buffer |

### Table 254: AMOUNTTX

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | AMOUNTTX | | | Number of bytes transmitted in last granted transaction |

### Table 255: CONFIG

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ORDER | | | Bit order |
| | | | MsbFirst | 0 | Most significant bit shifted out first |
| | | | LsbFirst | 1 | Least significant bit shifted out first |
| B | RW | CPHA | | | Serial clock (SCK) phase |
| | | | Leading | 0 | Sample on leading edge of clock, shift serial data on trailing edge |
| | | | Trailing | 1 | Sample on trailing edge of clock, shift serial data on leading edge |
| C | RW | CPOL | | | Serial clock (SCK) polarity |
| | | | ActiveHigh | 0 | Active high |
| | | | ActiveLow | 1 | Active low |

### Table 256: DEF

| Bit number | | | | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DEF | | | Default character. Character clocked out in case of an ignored transaction. |

### Table 257: ORC

| Bit number | | | | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ORC | | | Over-read character. Character clocked out after an over-read of the transmit buffer. |

# 28 I²C compatible Two Wire Interface (TWI)

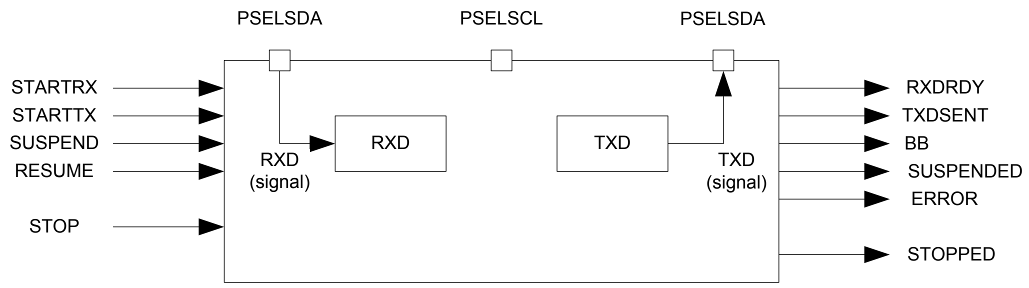



**Figure 62: TWI master's main features**

## 28.1 Functional description

The TWI master is compatible with I²C operating at 100 kHz and 400 kHz. This TWI master is not compatible with CBUS. As illustrated in *Figure 62: TWI master's main features* on page 144, the TWI transmitter and receiver are single buffered.

A TWI setup comprising one master and three slaves is illustrated in *Figure 63: A typical TWI setup comprising one master and three slaves* on page 144. This TWI master is only able to operate as the only master on the TWI bus.

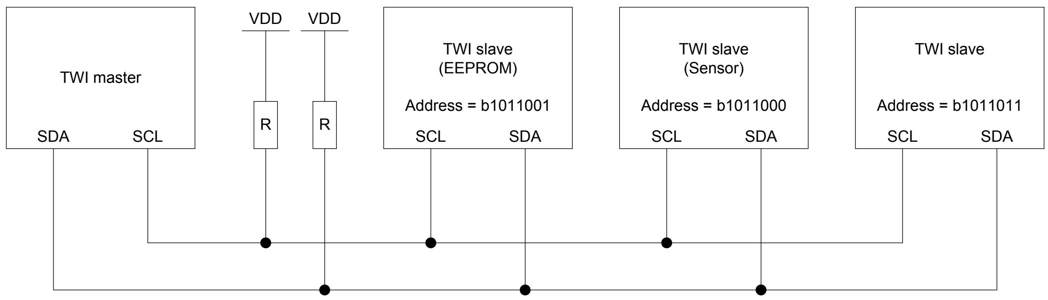



**Figure 63: A typical TWI setup comprising one master and three slaves**

This TWI master supports clock stretching performed by the slaves. The TWI master is started by triggering the STARTTX or STARTRX tasks, and stopped by triggering the STOP task.

If a NACK is clocked in from the slave, the TWI master will generate an ERROR event.

## 28.2 Master mode pin configuration

The different signals SCL and SDA associated with the TWI master are mapped to physical pins according to the configuration specified in the PSELSCL and PSELSDA registers respectively. If a value of 0xFFFFFFFF is specified in any of these registers, the associated TWI master signal is not connected to any physical pin. The PSELSCL and PSELSDA registers and their configurations are only used as long as the TWI master is enabled, and retained only as long as the device is in ON mode. PSELSCL and PSESDA must only be configured when the TWI is disabled.

To secure correct signal levels on the pins used by the TWI master when the system is in OFF mode, and when the TWI master is disabled, these pins must be configured in the GPIO peripheral as described in *Table 258: GPIO configuration* on page 145.

Only one peripheral can be assigned to drive a particular GPIO pin at a time, failing to do so may result in unpredictable behavior.

### Table 258: GPIO configuration

| TWI master signal | TWI master pin | Direction | Drive strength | Output value |
|---|---|---|---|---|
| SCL | As specified in PSELSCL | Input | S0D1 | Not applicable |
| SDA | As specified in PSELSDA | Input | S0D1 | Not applicable |

## 28.3 Shared resources

The TWI shares registers and other resources with other peripherals that have the same ID as the TWI. Therefore, you must disable all peripherals that have the same ID as the TWI before the TWI can be configured and used. Disabling a peripheral that has the same ID as the TWI will not reset any of the registers that are shared with the TWI. It is therefore important to configure all relevant TWI registers explicitly to secure that it operates correctly.

The Instantiation table in *Instantiation* on page 17 shows which peripherals have the same ID as the TWI.

## 28.4 Master write sequence

A TWI master write sequence is started by triggering the STARTTX task. After the STARTTX task has been triggered, the TWI master will generate a start condition on the TWI bus, followed by clocking out the address and the READ/WRITE bit set to 0 (WRITE=0, READ=1). The address must match the address of the slave device that the master wants to write to. The READ/WRITE bit is followed by an ACK/NACK bit (ACK=0 or NACK=1) generated by the slave.

After receiving the ACK bit, the TWI master will clock out the data bytes that are written to the TXD register. Each byte clocked out from the master will be followed by an ACK/NACK bit clocked in from the slave. A TXDSENT event will be generated each time the TWI master has clocked out a TXD byte, and the associated ACK/NACK bit has been clocked in from the slave.

The TWI master transmitter is single buffered, and a second byte can only be written to the TXD register after the previous byte has been clocked out and the ACK/NACK bit clocked in, that is, after the TXDSENT event has been generated.

If the CPU is prevented from writing to TXD when the TWI master is ready to clock out a byte, the TWI master will stretch the clock until the CPU has written a byte to the TXD register.

A typical TWI master write sequence is illustrated in *Figure 64: The TWI master writing data to a slave* on page 145. Occurrence 3 in *Figure 64: The TWI master writing data to a slave* on page 145 illustrates delayed processing of the TXDSENT event associated with TXD byte 1. In this scenario the TWI master will stretch the clock to prevent writing erroneous data to the slave.

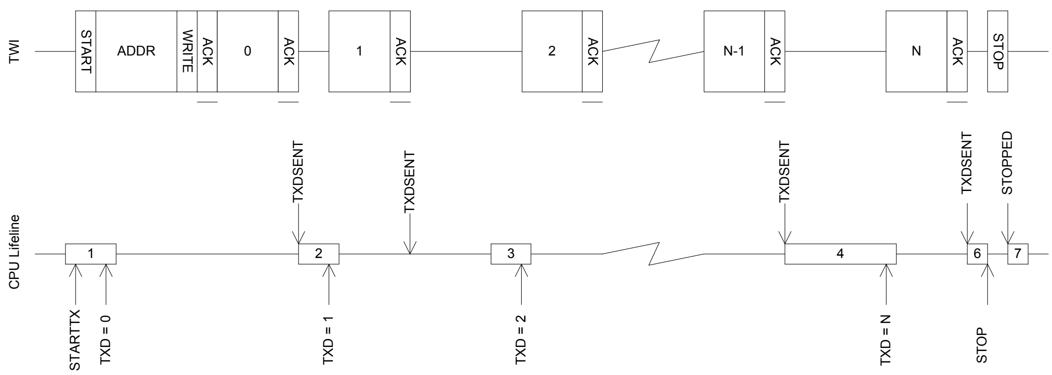



**Figure 64: The TWI master writing data to a slave**

The TWI master write sequence is stopped when the STOP task is triggered whereupon the TWI master will generate a stop condition on the TWI bus.

## 28.5 Master read sequence

A TWI master read sequence is started by triggering the STARTRX task. After the STARTRX task has been triggered the TWI master will generate a start condition on the TWI bus, followed by clocking out the address and the READ/WRITE bit set to 1 (WRITE = 0, READ = 1). The address must match the address of the slave device that the master wants to read from. The READ/WRITE bit is followed by an ACK/NACK bit (ACK=0 or NACK = 1) generated by the slave.

After having sent the ACK bit the TWI slave will send data to the master using the clock generated by the master.

The TWI master will generate a RXDRDY event every time a new byte is received in the RXD register.

After receiving a byte, the TWI master will delay sending the ACK/NACK bit by stretching the clock until the CPU has extracted the received byte, that is, by reading the RXD register.

The TWI master read sequence is stopped by triggering the STOP task. This task must be triggered before the last byte is extracted from RXD to ensure that the TWI master sends a NACK back to the slave before generating the stop condition.

A typical TWI master read sequence is illustrated in *Figure 65: The TWI master reading data from a slave* on page 146. Occurrence 3 in *Figure 65: The TWI master reading data from a slave* on page 146 illustrates delayed processing of the RXDRDY event associated with RXD byte B. In this scenario the TWI master will stretch the clock to prevent the slave from overwriting the contents of the RXD register.

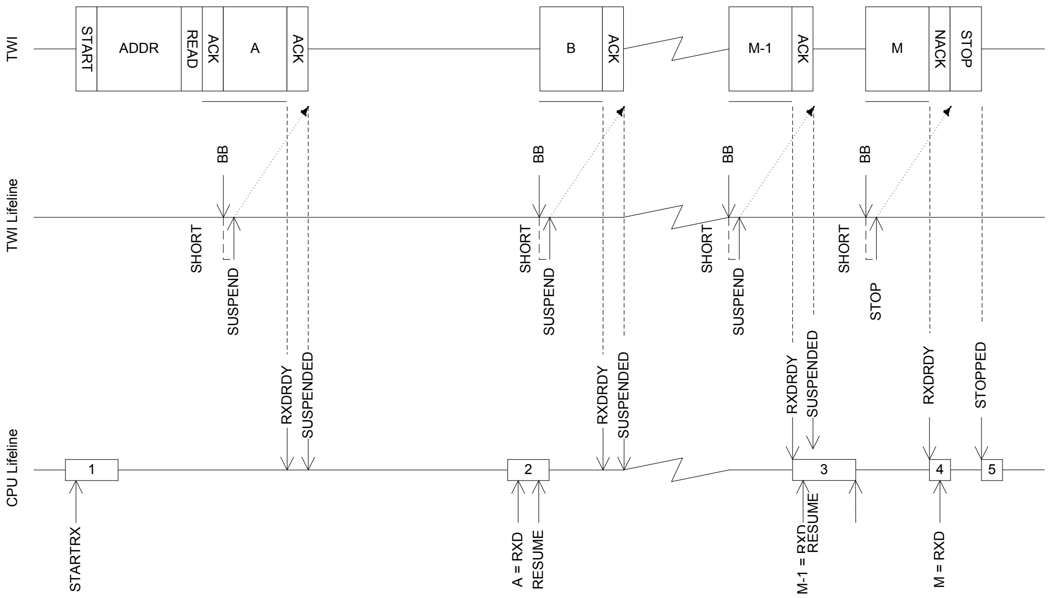



**Figure 65: The TWI master reading data from a slave**

## 28.6 Master repeated start sequence

*Figure 66: A repeated start sequence, where the TWI master writes one byte, followed by reading M bytes from the slave without performing a stop in-between* on page 147 illustrates a typical repeated start sequence where the TWI master writes one byte to the slave followed by reading M bytes from the slave. Any combination and number of transmit and receive sequences can be combined in this fashion. Only one shortcut to STOP can be enabled at any given time.

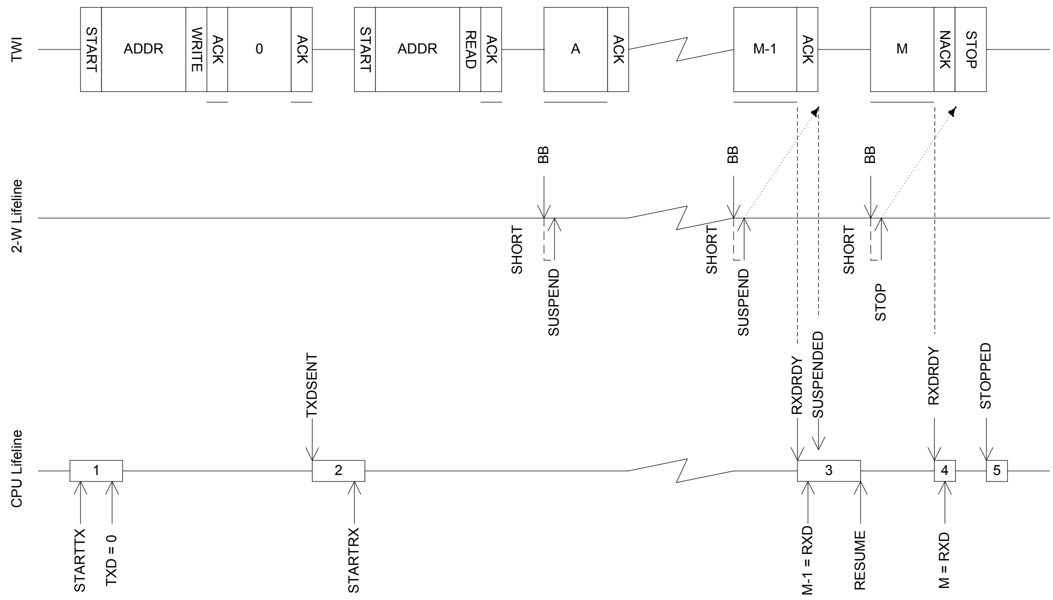



**Figure 66: A repeated start sequence, where the TWI master writes one byte, followed by reading M bytes from the slave without performing a stop in-between**

To generate a repeated start after a read sequence, a second start task must be triggered instead of the STOP task, that is, STARTRX or STARTTX. This start task must be triggered before the last byte is extracted from RXD to ensure that the TWI master sends a NACK back to the slave before generating the repeated start condition.

## 28.7 Register Overview

### Table 259: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40003000 | TWI | TWI0 | I2C compatible Two-Wire Interface |
| 0x40004000 | TWI | TWI1 | I2C compatible Two-Wire Interface |

### Table 260: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *STARTRX* | 0x000 | Start TWI receive sequence |
| *STARTTX* | 0x008 | Start TWI transmit sequence |
| *STOP* | 0x014 | Stop TWI transaction |
| *SUSPEND* | 0x01C | Suspend TWI transaction |
| *RESUME* | 0x020 | Resume TWI transaction |
| Events | | |
| *STOPPED* | 0x104 | TWI stopped |
| *RXDREADY* | 0x108 | TWI RXD byte received |
| *TXDSENT* | 0x11C | TWI TXD byte sent |
| *ERROR* | 0x124 | TWI error |
| *BB* | 0x138 | TWI byte boundary, generated before each byte that is sent or received |
| Registers | | |
| *SHORTS* | 0x200 | Shortcut register |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *ERRORSRC* | 0x4C4 | Error source |
| *ENABLE* | 0x500 | Enable TWI |
| *PSELSCL* | 0x508 | Pin select for SCL |
| *PSELSDA* | 0x50C | Pin select for SDA |
| *RXD* | 0x518 | RXD register |
| *TXD* | 0x51C | TXD register |
| *FREQUENCY* | 0x524 | TWI frequency |
| *ADDRESS* | 0x588 | Address used in the TWI transfer |

## 28.8 Register Details

### Table 261: SHORTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | BB_SUSPEND | | | Shortcut between *BB* event and *SUSPEND* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| B | RW | BB_STOP | | | Shortcut between *BB* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

### Table 262: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | STOPPED | | | Enable or disable interrupt on *STOPPED* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | RXDREADY | | | Enable or disable interrupt on *RXDREADY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| C | RW | TXDSENT | | | Enable or disable interrupt on *TXDSENT* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| D | RW | ERROR | | | Enable or disable interrupt on *ERROR* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| E | RW | BB | | | Enable or disable interrupt on *BB* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 263: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | STOPPED | | | Write '1' to Enable interrupt on *STOPPED* event. |
| | | | Enabled | 1 | Enable |
| B | RW | RXDREADY | | | Write '1' to Enable interrupt on *RXDREADY* event. |
| | | | Enabled | 1 | Enable |
| C | RW | TXDSENT | | | Write '1' to Enable interrupt on *TXDSENT* event. |
| | | | Enabled | 1 | Enable |
| D | RW | ERROR | | | Write '1' to Enable interrupt on *ERROR* event. |
| | | | Enabled | 1 | Enable |
| E | RW | BB | | | Write '1' to Enable interrupt on *BB* event. |
| | | | Enabled | 1 | Enable |

### Table 264: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | STOPPED | | | Write '1' to Clear interrupt on *STOPPED* event. |
| | | | Disabled | 1 | Disable |
| B | RW | RXDREADY | | | Write '1' to Clear interrupt on *RXDREADY* event. |
| | | | Disabled | 1 | Disable |
| C | RW | TXDSENT | | | Write '1' to Clear interrupt on *TXDSENT* event. |
| | | | Disabled | 1 | Disable |
| D | RW | ERROR | | | Write '1' to Clear interrupt on *ERROR* event. |
| | | | Disabled | 1 | Disable |
| E | RW | BB | | | Write '1' to Clear interrupt on *BB* event. |
| | | | Disabled | 1 | Disable |

## Table 265: ERRORSRC

**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OVERRUN | | | Overrun error<br>A new byte was received before previous byte got read by software from the RXD register. (Previous data is lost) |
| | | | NotPresent | 0 | Read: no overrun occured |
| | | | Present | 1 | Read: overrun occured |
| | | | Clear | 1 | Write: clear error on writing '1' |
| B | RW | ANACK | | | NACK received after sending the address (write '1' to clear) |
| | | | NotPresent | 0 | Read: error not present |
| | | | Present | 1 | Read: error present |
| | | | Clear | 1 | Write: clear error on writing '1' |
| C | RW | DNACK | | | NACK received after sending a data byte (write '1' to clear) |
| | | | NotPresent | 0 | Read: error not present |
| | | | Present | 1 | Read: error present |
| | | | Clear | 1 | Write: clear error on writing '1' |

## Table 266: 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 | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | Enable or disable TWI |
| | | | Disabled | 0 | Disable TWI |
| | | | Enabled | 5 | Enable TWI |

## Table 267: PSELSCL

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELSCL | | [0..31] | Pin number configuration for TWI SCL signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 268: PSELSDA

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELSDA | | [0..31] | Pin number configuration for TWI SDA signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 269: RXD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RXD | | | RXD register |

## Table 270: TXD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TXD | | | TXD register |

## Table 271: 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 | 0 | 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 | 0 | 0 | 0 | 0 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | FREQUENCY | | | TWI master clock frequency |
| | | | K100 | 0x01980000 | 100 kbps |
| | | | K250 | 0x04000000 | 250 kbps |
| | | | K400 | 0x06680000 | 400 kbps (actual rate 410.256 kbps) |

### Table 272: 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ADDRESS | | | Address used in the TWI transfer |

# 29 Universal Asynchronous Receiver/Transmitter (UART)

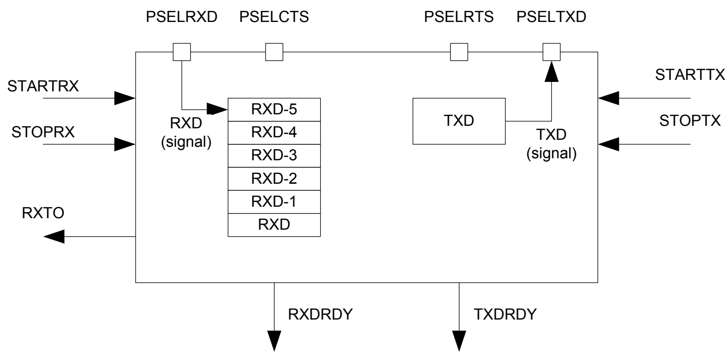



**Figure 67: UART configuration**

## 29.1 Functional description

The UART implements support for the following features:

- Full-duplex operation
- Automatic flow control
- Parity checking and generation for the 9th data bit

As illustrated in *Figure 67: UART configuration* on page 151, the UART uses the TXD and RXD registers directly to transmit and receive data. The UART uses one stop bit.

## 29.2 Pin configuration

The different signals RXD, CTS (Clear To Send, active low), RTS (Request To Send, active low), and TXD associated with the UART are mapped to physical pins according to the configuration specified in the PSELRXD, PSELCTS, PSELRTS, and PSELTXD registers respectively. If a value of 0xFFFFFFFF is specified in any of these registers, the associated UART signal will not be connected to any physical pin. The PSELRXD, PSELCTS, PSELRTS, and PSELTXD registers and their configurations are only used as long as the UART is enabled, and retained only for the duration the device is in ON mode. PSELRXD, PSELRTS, PSELTRTS and PSELTXD must only be configured when the UART is disabled.

To secure correct signal levels on the pins by the UART when the system is in OFF mode, the pins must be configured in the GPIO peripheral as described in *Table 273: GPIO configuration* on page 151.

Only one peripheral can be assigned to drive a particular GPIO pin at a time. Failing to do so may result in unpredictable behavior.

**Table 273: GPIO configuration**

| UART pin | Direction | Output value |
|---|---|---|
| RXD | Input | Not applicable |
| CTS | Input | Not applicable |
| RTS | Output | 1 |
| TXD | Output | 1 |

## 29.3 Shared resources

The UART shares registers and other resources with other peripherals that have the same ID as the UART. Therefore, you must disable all peripherals that have the same ID as the UART before the UART can be configured and used.

Disabling a peripheral that has the same ID as the UART will not reset any of the registers that are shared with the UART. It is therefore important to configure all relevant UART registers explicitly to ensure that it operates correctly.

See the Instantiation table in *Instantiation* on page 17 for details on peripherals and their IDs.

## 29.4 Transmission

A UART transmission sequence is started by triggering the STARTTX task. Bytes are transmitted by writing to the TXD register. When a byte has been successfully transmitted the UART will generate a TXDRDY event after which a new byte can be written to the TXD register. A UART transmission sequence is stopped immediately by triggering the STOPTX task.

If flow control is enabled a transmission will be automatically suspended when CTS is deactivated and resumed when CTS is activated again, as illustrated in *Figure 68: UART transmission* on page 152. A byte that is in transmission when CTS is deactivated will be fully transmitted before the transmission is suspended.

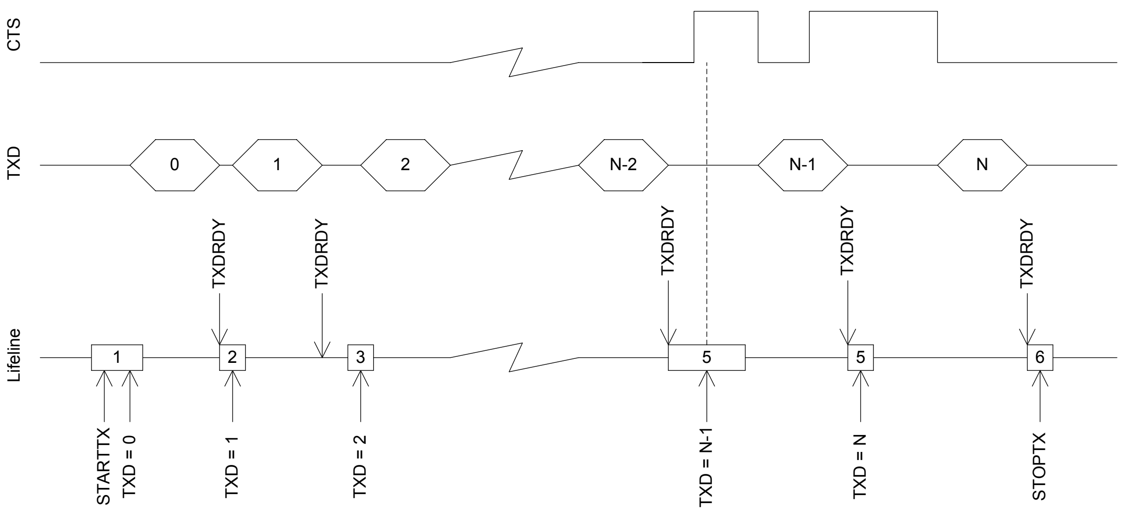



**Figure 68: UART transmission**

## 29.5 Reception

A UART reception sequence is started by triggering the STARTRX task. The UART receiver chain implements a FIFO capable of storing six incoming RXD bytes before data is overwritten. Bytes are extracted from this FIFO by reading the RXD register. When a byte is extracted from the FIFO a new byte pending in the FIFO will be moved to the RXD register. The UART will generate an RXDRDY event every time a new byte is moved to the RXD register.

When flow control is enabled, the UART will deactivate the RTS signal when there is only space for four more bytes in the receiver FIFO. The counterpart transmitter is therefore able to send up to four bytes after the RTS signal is deactivated before data is being overwritten. To prevent overwriting data in the FIFO, the counterpart UART transmitter must therefore make sure to stop transmitting data within four bytes after the RTS line is deactivated.

The RTS signal will first be activated again when the FIFO has been emptied, that is, when all bytes in the FIFO have been read by the CPU, see *Figure 69: UART reception* on page 153.

The RTS signal will also be deactivated when the receiver is stopped through the STOPRX task as illustrated in *Figure 69: UART reception* on page 153. The UART will be able to receive up to four bytes if they are sent in succession immediately after the RTS signal has been deactivated. This is possible because the UART is, even after the STOPRX task is triggered, able to receive bytes for an extended period equal to the time it takes to send four bytes on the configured baud rate. The UART will generate a receiver timeout event (RXTO) when this period has elapsed.

To prevent loss of incoming data the RXD register must only be read one time following every RXDRDY event.

To secure that the CPU can detect all incoming RXDRDY events through the RXDRDY event register, the RXDRDY event register must be cleared before the RXD register is read. The reason for this is that the UART is allowed to write a new byte to the RXD register, and therefore can also generate a new event, immediately after the RXD register is read (emptied) by the CPU.

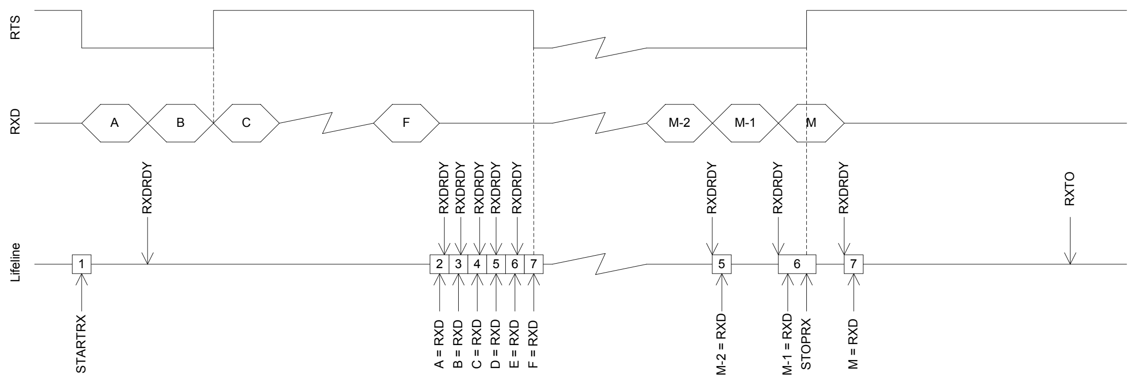



**Figure 69: UART reception**

As indicated in occurrence 2 in *Figure 69: UART reception* on page 153, the RXDRDY event associated with byte B is generated first after byte A has been extracted from RXD.

## 29.6 Suspending the UART

The UART can be suspended by triggering the SUSPEND task. SUSPEND will affect both the UART receiver and the UART transmitter, i.e. the transmitter will stop transmitting and the receiver will stop receiving. UART transmission and reception can be resumed, after being suspended, by triggering STARTTX and STARTRX respectively.

Following a SUSPEND task, an ongoing TXD byte transmission will be completed before the UART is suspended.

When the SUSPEND task is triggered, the UART receiver will behave in the same way as it does when the STOPRX task is triggered.

## 29.7 Error conditions

An ERROR event, in the form of a framing error, will be generated if a valid stop bit is not detected in a frame. Another ERROR event, in the form of a break condition, will be generated if the RXD line is held active low for longer than the length of a data frame. Effectively, a framing error is always generated before a break condition occurs.

## 29.8 Using the UART without flow control

If flow control is not enabled the interface will behave as if the CTS and RTS lines are kept active all the time.

## 29.9 Parity configuration

When parity is enabled, the parity will be generated automatically from the even parity of TXD and RXD for transmission and reception respectively.

## 29.10 Register Overview

### Table 274: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40002000 | UART | UART0 | Universal Asynchronous Receiver/Transmitter |

### Table 275: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *STARTRX* | 0x000 | Start UART receiver |
| *STOPRX* | 0x004 | Stop UART receiver |
| *STARTTX* | 0x008 | Start UART transmitter |
| *STOPTX* | 0x00C | Stop UART transmitter |
| *SUSPEND* | 0x01C | Suspend UART |
| Events | | |
| *CTS* | 0x100 | CTS is activated (set low). Clear To Send. |
| *NCTS* | 0x104 | CTS is deactivated (set high). Not Clear To Send. |
| *RXDRDY* | 0x108 | Data received in RXD |
| *TXDRDY* | 0x11C | Data sent from TXD |
| *ERROR* | 0x124 | Error detected |
| *RXTO* | 0x144 | Receiver timeout |
| Registers | | |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *ERRORSRC* | 0x480 | Error source |
| *ENABLE* | 0x500 | Enable UART |
| *PSELRTS* | 0x508 | Pin select for RTS |
| *PSELTXD* | 0x50C | Pin select for TXD |
| *PSELCTS* | 0x510 | Pin select for CTS |
| *PSELRXD* | 0x514 | Pin select for RXD |
| *RXD* | 0x518 | RXD register |
| *TXD* | 0x51C | TXD register |
| *BAUDRATE* | 0x524 | Baud rate |
| *CONFIG* | 0x56C | Configuration of parity and hardware flow control |

## 29.11 Register Details

### Table 276: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CTS | | | Enable or disable interrupt on *CTS* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | NCTS | | | Enable or disable interrupt on *NCTS* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| C | RW | RXDRDY | | | Enable or disable interrupt on *RXDRDY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| D | RW | TXDRDY | | | Enable or disable interrupt on *TXDRDY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| E | RW | ERROR | | | Enable or disable interrupt on *ERROR* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| F | RW | RXTO | | | Enable or disable interrupt on *RXTO* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 277: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CTS | | | Write '1' to Enable interrupt on *CTS* event. |
| | | | Enabled | 1 | Enable |
| B | RW | NCTS | | | Write '1' to Enable interrupt on *NCTS* event. |
| | | | Enabled | 1 | Enable |
| C | RW | RXDRDY | | | Write '1' to Enable interrupt on *RXDRDY* event. |
| | | | Enabled | 1 | Enable |
| D | RW | TXDRDY | | | Write '1' to Enable interrupt on *TXDRDY* event. |
| | | | Enabled | 1 | Enable |
| E | RW | ERROR | | | Write '1' to Enable interrupt on *ERROR* event. |
| | | | Enabled | 1 | Enable |
| F | RW | RXTO | | | Write '1' to Enable interrupt on *RXTO* event. |
| | | | Enabled | 1 | Enable |

## Table 278: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | CTS | | | Write '1' to Clear interrupt on *CTS* event. |
| | | | Disabled | 1 | Disable |
| B | RW | NCTS | | | Write '1' to Clear interrupt on *NCTS* event. |
| | | | Disabled | 1 | Disable |
| C | RW | RXDRDY | | | Write '1' to Clear interrupt on *RXDRDY* event. |
| | | | Disabled | 1 | Disable |
| D | RW | TXDRDY | | | Write '1' to Clear interrupt on *TXDRDY* event. |
| | | | Disabled | 1 | Disable |
| E | RW | ERROR | | | Write '1' to Clear interrupt on *ERROR* event. |
| | | | Disabled | 1 | Disable |
| F | RW | RXTO | | | Write '1' to Clear interrupt on *RXTO* event. |
| | | | Disabled | 1 | Disable |

## Table 279: ERRORSRC

**Note:** Individual bits are cleared by writing a '1' to the bits that shall be cleared. Writing a '0' will have no effect.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | OVERRUN | | | Overrun error<br>A start bit is received while the previous data still lies in RXD. (Previous data is lost.) |
| | | | NotPresent | 0 | Read: error not present |
| | | | Present | 1 | Read: error present |
| | | | Clear | 1 | Write: clear error on writing '1' |
| B | RW | PARITY | | | Parity error<br>A character with bad parity is received, if HW parity check is enabled. |
| | | | NotPresent | 0 | Read: error not present |
| | | | Present | 1 | Read: error present |
| | | | Clear | 1 | Write: clear error on writing '1' |
| C | RW | FRAMING | | | Framing error occurred<br>A valid stop bit is not detected on the serial data input after all bits in a character have been received. |
| | | | NotPresent | 0 | Read: error not present |
| | | | Present | 1 | Read: error present |
| | | | Clear | 1 | Write: clear error on writing '1' |
| D | RW | BREAK | | | Break condition<br>The serial data input is '0' for longer than the length of a data frame. (The data frame length is 10 bits without parity bit, and 11 bits with parity bit.). |
| | | | NotPresent | 0 | Read: error not present |
| | | | Present | 1 | Read: error present |
| | | | Clear | 1 | Write: clear error on writing '1' |

## Table 280: 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 | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | Enable or disable UART |
| | | | Disabled | 0 | Disable UART |
| | | | Enabled | 4 | Enable UART |

## Table 281: PSELRTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELRTS | | [0..31] | Pin number configuration for UART RTS signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 282: PSELTXD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELTXD | | [0..31] | Pin number configuration for UART TXD signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 283: PSELCTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELCTS | | [0..31] | Pin number configuration for UART CTS signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 284: PSELRXD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELRXD | | [0..31] | Pin number configuration for UART RXD signal |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 285: RXD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RXD | | | RX data received in previous transfers, double buffered |

## Table 286: TXD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | TXD | | | TX data to be transferred |

## Table 287: BAUDRATE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 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 | 0 | 0 | 0 | 0 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | BAUDRATE | | | Baud-rate |
| | | | Baud1200 | 0x0004F000 | 1200 baud (actual rate: 1205) |
| | | | Baud2400 | 0x0009D000 | 2400 baud (actual rate: 2396) |
| | | | Baud4800 | 0x0013B000 | 4800 baud (actual rate: 4808) |
| | | | Baud9600 | 0x00275000 | 9600 baud (actual rate: 9598) |
| | | | Baud14400 | 0x003B0000 | 14400 baud (actual rate: 14414) |
| | | | Baud19200 | 0x004EA000 | 19200 baud (actual rate: 19208) |
| | | | Baud28800 | 0x0075F000 | 28800 baud (actual rate: 28829) |
| | | | Baud31250 | 0x00800000 | 31250 baud |
| | | | Baud38400 | 0x009D5000 | 38400 baud (actual rate: 38462) |
| | | | Baud56000 | 0x00E50000 | 56000 baud (actual rate: 55944) |
| | | | Baud57600 | 0x00EBF000 | 57600 baud (actual rate: 57602) |
| | | | Baud76800 | 0x013A9000 | 76800 baud (actual rate: 76797) |
| | | | Baud115200 | 0x01D7E000 | 115200 baud (actual rate: 115204) |
| | | | Baud230400 | 0x03AFB000 | 230400 baud (actual rate: 230393) |
| | | | Baud250000 | 0x04000000 | 250000 baud |
| | | | Baud460800 | 0x075F7000 | 460800 baud |
| | | | Baud921600 | 0x0EBEDFA4 | 921600 baud (actual rate: 921585) |
| | | | Baud1M | 0x10000000 | 1Mega baud |

## Table 288: CONFIG

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | HWFC | | | Hardware flow control |
| | | | Disabled | 0 | Disabled |
| | | | Enabled | 1 | Enabled |
| B | RW | PARITY | | | Parity |
| | | | Excluded | 0x0 | Exclude parity bit |
| | | | Included | 0x7 | Include parity bit |

# 30 Quadrature Decoder (QDEC)

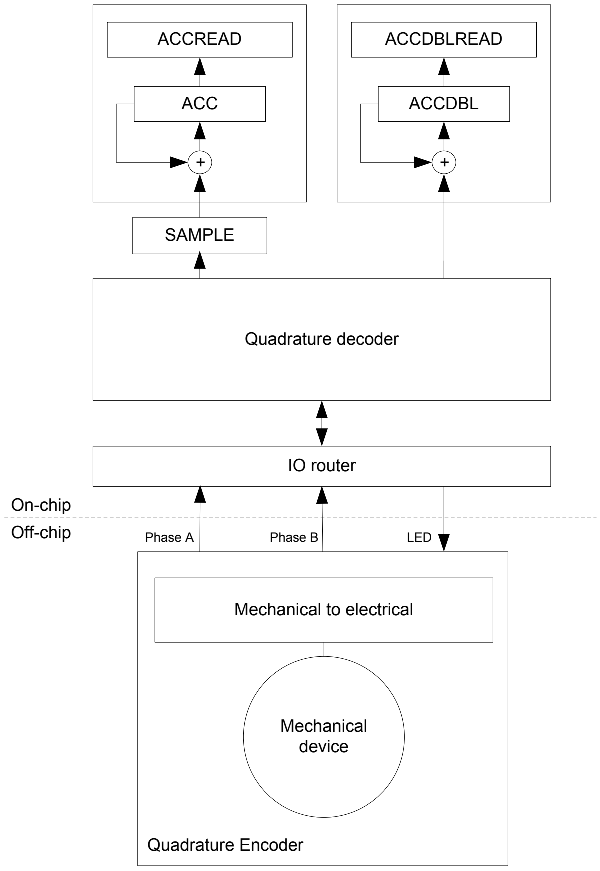



**Figure 70: Quadrature decoder configuration**

## 30.1 Functional description

The Quadrature Decoder (QDEC) can be used for decoding the output of an off-chip quadrature encoder. The QDEC provides the following:

- Decoding of digital waveform from off-chip quadrature encoder.
- Sample accumulation eliminating hard real-time requirements to be enforced on application.
- Optional input debounce filters.
- Optional LED output signal for optical encoders.

### 30.1.1 Pin configuration

The different signals: Phase A, Phase B, and LED, are mapped to physical pins according to the configuration specified in the PSELA, PSELB, and PSELLED registers respectively. If a value of 0xFFFFFFFF is specified in any of these registers, the associated signal will not be connected to any physical pin. The PSELA, PSELB, and PSELLED registers and their configurations are only used as long as the QDEC is enabled, and retained only as long as the device is in ON mode.

To secure correct behavior in the QDEC, the pins used by the QDEC must be configured in the GPIO peripheral as described in *Table 289: GPIO configuration* on page 158 prior to enabling the QDEC. This configuration must be retained in the GPIO for the selected IOs as long as the QDEC is enabled.

Only one peripheral can be assigned to drive a particular GPIO pin at a time, failing to do so may result in unpredictable behavior.

### Table 289: GPIO configuration

| QDEC signal | QDEC pin | Direction | Output value |
|---|---|---|---|
| Phase A | As specified in PSELA | Input | Not applicable |
| Phase B | As specified in PSELB | Input | Not applicable |
| LED | As specified in PSELLED | Input | Not applicable |

## 30.1.2 Sampling and decoding

The off-chip quadrature encoder is an incremental motion encoder outputting two waveforms; phase A and phase B. The two output waveforms are always 90 degrees out of phase, meaning that one always changes level before the other. The direction of movement is indicated by which of these two waveforms that changes level first. Invalid transitions may occur, that is when the two waveforms switch simultaneously. This may occur if the wheel rotates too fast relative to the sample rate set for the decoder.

The QDEC decodes the output from the off-chip encoder by sampling the QDEC phase input pins (A and B) at a fixed rate as specified in the SAMPLEPER register.

When started, the decoder continuously samples the two input waveforms and decodes these by comparing the current sample pair (n) with the previous sample pair (n-1).

The decoding of the sample pairs is described in *Table 290: Sampled value encoding* on page 158.

### Table 290: Sampled value encoding

| Previous sample pair(n - 1) | | Current samples pair(n) | | SAMPLE register | ACC operation | ACCDBL operation | Description |
|---|---|---|---|---|---|---|---|
| A | B | A | B | | | | |
| 0 | 0 | 0 | 0 | 0 | No change | No change | No movement |
| 0 | 0 | 0 | 1 | 1 | Increment | No change | Movement in positive direction |
| 0 | 0 | 1 | 0 | -1 | Decrement | No change | Movement in negative direction |
| 0 | 0 | 1 | 1 | 2 | No change | Increment | Error: Double transition |
| 0 | 1 | 0 | 0 | -1 | Decrement | No change | Movement in negative direction |
| 0 | 1 | 0 | 1 | 0 | No change | No change | No movement |
| 0 | 1 | 1 | 0 | 2 | No change | Increment | Error: Double transition |
| 0 | 1 | 1 | 1 | 1 | Increment | No change | Movement in positive direction |
| 1 | 0 | 0 | 0 | 1 | Increment | No change | Movement in positive direction |
| 1 | 0 | 0 | 1 | 2 | No change | Increment | Error: Double transition |
| 1 | 0 | 1 | 0 | 0 | No change | No change | No movement |
| 1 | 0 | 1 | 1 | -1 | Decrement | No change | Movement in negative direction |
| 1 | 1 | 0 | 0 | 2 | No change | Increment | Error: Double transition |
| 1 | 1 | 0 | 1 | -1 | Decrement | No change | Movement in negative direction |
| 1 | 1 | 1 | 0 | 1 | Increment | No change | Movement in positive direction |
| 1 | 1 | 1 | 1 | 0 | No change | No change | No movement |

## 30.1.3 LED output

The LED output follows the sample period and the LED is switched on a given period prior to sampling and switched off immediately after the inputs are sampled. The period the LED is switched on prior to sampling is given in the LEDPRE register.

The LED output pin polarity is specified in the LEDPOL register.

For using off-chip mechanical encoders not requiring a LED, the LED output can be disabled by writing 0xFFFFFFFF to the PSELLED register. In this case the QDEC will not acquire access to a LED output pin and the pin can be used for other purposes by the CPU.

## 30.1.4 Debounce filters

Each of the two phase inputs have digital debounce filters. When enabled through the DBFEN register, the filter inputs are sampled at a fixed 1 MHz frequency during the entire sample period (which is specified in the SAMPLEPER register), and the filters require all of the samples within this sample period to equal before the input signal is accepted and transferred to the output of the filter.

As a result, only input signal with a steady state longer than twice the period specified in SAMPLEPER are guaranteed to pass through the filter, and any signal with a steady state shorter than SAMPLEPER will always be suppressed by the filter. (This is assumed that the frequency during the debounce period never exceeds 500 kHz (as required by the Nyquist theorem when using a 1 MHz sample frequency).

> **Note:** The LED will always be ON when the debounce filters are enabled, as the inputs in this case will be sampled continuously.

### 30.1.5 Accumulators

The quadrature decoder contains two accumulator registers, ACC and ACCDBL, that accumulate respectively valid motion sample values and the number of detected invalid samples (double transitions).

The ACC register will accumulate all valid values (1/-1) written to the SAMPLE register. This can be useful for preventing hard real-time requirements from being enforced on the application. When using the ACC register the application does not need to read every single sample from the SAMPLE register, but can instead fetch the ACC register whenever it fits the application. The ACC register will always hold the relative movement of the external mechanical device since the previous clearing of the ACC register. Sample values indicating a double transition (2) will not be accumulated in the ACC register.

An ACCOF event will be generated if the ACC receives a SAMPLE value that would cause the register to overflow or underflow. Any SAMPLE value that would cause an ACC overflow or underflow will be discarded, but any samples not causing the ACC to overflow or underflow will still be accepted.

The accumulator ACCDBL accumulates the number of detected double transitions since the previous clearing of the ACCDBL register.

The ACC and ACCDBL registers can be cleared by the READCLRACC and subsequently read using the ACCREAD and ACCDBLREAD registers.

The REPORTPER register allows automating the capture of several samples before sending out a REPORTRDY event in case a non-null displacement has been captured and accumulated. The REPORTPER field in this register selects after how many samples the accumulator content is evaluated to send (or not) a REPORTRDY event.

### 30.1.6 Output/input pins

The QDEC uses a 3 pin interface to the off-chip quadrature encoder.

These pins will be acquired when the QDEC is enabled in the ENABLE register. The pins acquired by the QDEC cannot be written by the CPU, but they can still be read by the CPU.

The pin numbers to be used for the QDEC are selected using the PSELn registers.

## 30.2 Register Overview

#### Table 291: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40012000 | QDEC | QDEC | Quadrature Decoder |

#### Table 292: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *START* | 0x000 | Task starting the quadrature decoder |
| *STOP* | 0x004 | Task stopping the quadrature decoder |
| *READCLRACC* | 0x008 | Read and clear ACC and ACCDBL |
| Events | | |
| *SAMPLERDY* | 0x100 | Event being generated for every new sample value written to the SAMPLE register |
| *REPORTRDY* | 0x104 | Non-null report ready |
| *ACCOF* | 0x108 | ACC or ACCDBL register overflow |
| Registers | | |
| *SHORTS* | 0x200 | Shortcut register |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *ENABLE* | 0x500 | Enable the quadrature decoder |
| *LEDPOL* | 0x504 | LED output pin polarity |
| *SAMPLEPER* | 0x508 | Sample period |
| *SAMPLE* | 0x50C | Motion sample value |
| *REPORTPER* | 0x510 | Number of samples to be taken before a REPORTRDY event is generated |

| Register | Offset | Description |
|---|---|---|
| *ACC* | 0x514 | Register accumulating the valid transitions |
| *ACCREAD* | 0x518 | Snapshot of the ACC register, updated by the READCLRACC task |
| *PSELLED* | 0x51C | GPIO pin number to be used as LED output |
| *PSELA* | 0x520 | GPIO pin number to be used as Phase A input |
| *PSELB* | 0x524 | GPIO pin number to be used as Phase B input |
| *DBFEN* | 0x528 | Enable input debounce filters |
| *LEDPRE* | 0x540 | Time period the LED is switched ON prior to sampling |
| *ACCDBL* | 0x544 | Register accumulating the number of detected double transitions |
| *ACCDBLREAD* | 0x548 | Snapshot of the ACCDBL, updated by the READCLRACC task |

# 30.3 Register Details

## Table 293: SHORTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | REPORTRDY_READCLRACC | | | Shortcut between *REPORTRDY* event and *READCLRACC* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| B | RW | SAMPLERDY_STOP | | | Shortcut between *SAMPLERDY* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

## Table 294: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | SAMPLERDY | | | Enable or disable interrupt on *SAMPLERDY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | REPORTRDY | | | Enable or disable interrupt on *REPORTRDY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| C | RW | ACCOF | | | Enable or disable interrupt on *ACCOF* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

## Table 295: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | SAMPLERDY | | | Write '1' to Enable interrupt on *SAMPLERDY* event. |
| | | | Enabled | 1 | Enable |
| B | RW | REPORTRDY | | | Write '1' to Enable interrupt on *REPORTRDY* event. |
| | | | Enabled | 1 | Enable |
| C | RW | ACCOF | | | Write '1' to Enable interrupt on *ACCOF* event. |
| | | | Enabled | 1 | Enable |

## Table 296: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | SAMPLERDY | | | Write '1' to Clear interrupt on *SAMPLERDY* event. |
| | | | Disabled | 1 | Disable |
| B | RW | REPORTRDY | | | Write '1' to Clear interrupt on *REPORTRDY* event. |
| | | | Disabled | 1 | Disable |
| C | RW | ACCOF | | | Write '1' to Clear interrupt on *ACCOF* event. |
| | | | Disabled | 1 | Disable |

## Table 297: 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | Enable or disable the quadrature 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
|  |  |  |  |  | When enabled the decoder pins will be active. When disabled the quadrature decoder pins is not active and can be used as GPIO . |
|  |  |  | Disabled | 0 | Disable |
|  |  |  | Enabled | 1 | Enable |

## Table 298: LEDPOL

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | LEDPOL |  |  | LED output pin polarity |
|  |  |  | ActiveLow | 0 | Led active on output pin low |
|  |  |  | ActiveHigh | 1 | Led active on output pin high |

## Table 299: SAMPLEPER

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | SAMPLEPER |  |  | Sample period. The SAMPLE register will be updated for every new sample |
|  |  |  | 128us | 0 | 128 us |
|  |  |  | 256us | 1 | 256 us |
|  |  |  | 512us | 2 | 512 us |
|  |  |  | 1024us | 3 | 1024 us |
|  |  |  | 2048us | 4 | 2048 us |
|  |  |  | 4096us | 5 | 4096 us |
|  |  |  | 8192us | 6 | 8192 us |
|  |  |  | 16384us | 7 | 16384 us |

## Table 300: SAMPLE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | SAMPLE |  | [-1..2] | Last motion sample<br>The value is a 2's complement value, and the sign gives the direction of the motion. The value '2' indicates a double transition. |

## Table 301: REPORTPER

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | REPORTPER |  |  | Specifies the number of samples to be accumulated in the ACC register before the REPORTRDY event can be generated<br>The report period in [us] is given as: RPUS = SP * RP Where RPUS is the report period in [us/report] SP is the sample period in [us/sample] specified in SAMPLEPERRP is the report period in [samples/report] specified in REPORTPER . |
|  |  |  | 10Smpl | 0 | 10 samples / report |
|  |  |  | 40Smpl | 1 | 40 samples / report |
|  |  |  | 80Smpl | 2 | 80 samples / report |
|  |  |  | 120Smpl | 3 | 120 samples / report |
|  |  |  | 160Smpl | 4 | 160 samples / report |
|  |  |  | 200Smpl | 5 | 200 samples / report |
|  |  |  | 240Smpl | 6 | 240 samples / report |
|  |  |  | 280Smpl | 7 | 280 samples / report |

## Table 302: ACC

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | ACC |  | [-1024..1023] | Register accumulating all valid samples (not double transition) read from the RENC (in the SAMPLE register)<br>Double transitions ( SAMPLE = 2 ) will not be accumulated in this register. The value is a 32 bit 2's complement value. If a sample that would cause this register to overflow or underflow |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | | | is received, the sample will be ignored and an overflow event ( ACCOF ) will be generated. The ACC register is cleared by triggering the READCLRACC task. |

## Table 303: ACCREAD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | ACCREAD | | [-1024..1023] | Snapshot of the ACC register.<br>The ACCREAD register is updated when the READCLRACC task is triggered |

## Table 304: PSELLED

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELLED | | [0..31] | GPIO pin number to be used as LED output. Writing the value 0xFFFFFFFF will disable this output. |

## Table 305: PSELA

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELA | | [0..31] | GPIO pin number to be used as Phase A input. Writing the value 0xFFFFFFFF will disable this input. |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 306: PSELB

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 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 | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | PSELB | | [0..31] | GPIO pin number to be used as Phase B input. Writing the value 0xFFFFFFFF will disable this input. |
| | | | Disconnected | 0xFFFFFFFF | Disconnect |

## Table 307: DBFEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | DBFEN | | | Enable input debounce filters |
| | | | Disabled | 0 | Debounce input filters disabled |
| | | | Enabled | 1 | Debounce input filters enabled |

## Table 308: LEDPRE

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 |
| Reset | 0 | 0 | 0 | 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 | 0 | 0 | 0 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | LEDPRE | | [0..511] | Period in us the LED is switched on prior to sampling |

## Table 309: ACCDBL

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | ACCDBL | | [0..15] | Register accumulating the number of detected double or illegal transitions. ( SAMPLE = 2 ).<br>When this register has reached its maximum value the accumulation of double / illegal transitions will stop. An overflow event ( ACCOF ) will be generated if any double or illegal transitions are detected after the maximum value was |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | | | reached. This field is cleared by triggering the READCLRACC task. |

## Table 310: ACCDBLREAD

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | ACCDBLREAD | | [0..15] | Snapshot of the ACCDBL register. This field is updated when the READCLRACC task is triggered. |

# 31 Analog to Digital Converter (ADC)

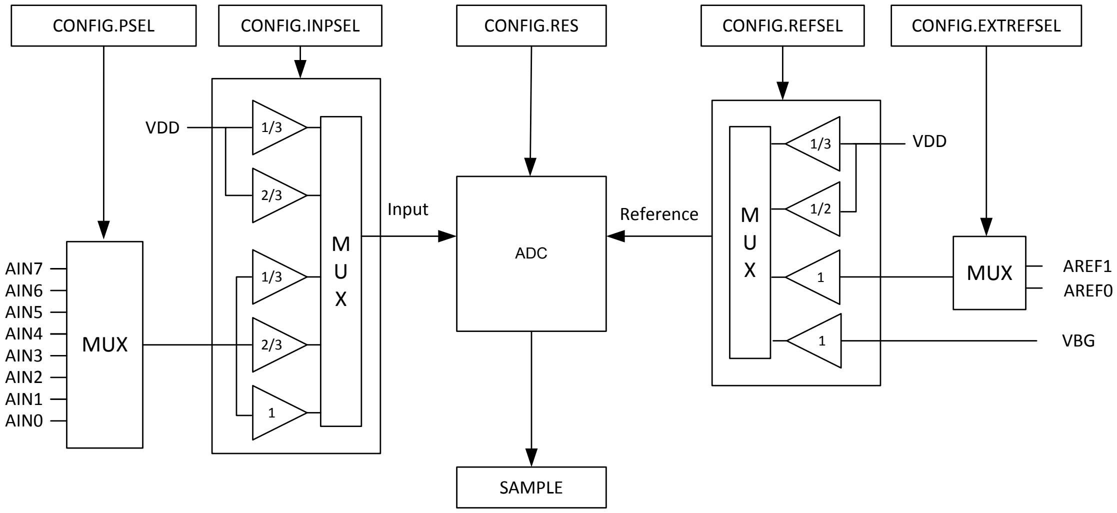



**Figure 71: Analog to digital converter**

## 31.1 Functional description

### 31.1.1 Usage

An ADC conversion is started by using the START task, either by writing the task register directly from the CPU or by triggering the task through the PPI.

During sampling the ADC will enter a busy state. The ADC busy/ready state can be monitored via the BUSY register.

When the ADC conversion is completed, an END event will be generated and the result of the conversion can be read from the RESULT register.

When the ADC conversion is completed, the ADC analog electronics power down to save power.

### 31.1.2 One-shot / continuous operation

The ADC itself only supports one-shot operation, this means every single conversion has to be explicitly started using the START task.

However, continuous ADC operation can be achieved by continuously triggering the START task from, for example, a timer through the PPI.

### 31.1.3 Configuration

All parameters such as input selection, reference selection, resolution, pre-scaling etc. are configured using the CONFIG register. It is important to configure all relevant ADC registers explicitly to secure that it operates correctly.

The user can use the PSEL register to select one of the analog input pins, AIN0 through AIN7, as input for the ADC. See the device product specification for more information about which analog pins are available on a particular device. The selected analog pin will be acquired by the ADC when it is enabled through the ENABLE register, see *GPIO* chapter for more information on how analog pins are selected.

> **Note:** It is not allowed to configure the ADC during an on-going ADC conversion (ADC busy).

### 31.1.4 Input voltage range

It is very important you configure the ADC so the input voltage range and the ADC voltage range is matching.

If the input voltage range is lower than the ADC voltage range, the resolution will not be fully utilized.

If the input voltage range is higher than the ADC voltage range, all values above the maximum ADC voltage range will be limited to the maximum value, also called the saturation point.

Input voltage range and saturation point depends on the configured ADC reference voltage and the chosen prescaling. If the 1.2 V VBG internal reference voltage is used with 1/1 prescaling, the ADC range will be 0-1.2 V with a saturation point of 1.2 V. This means that your AIN signal with 1/1 prescaling should be in the range 0-1.2 V in order to obtain proper conversion. Input above 1.2 V will be converted to the maximum ADC value. However, if you use, for example, 1/3 prescaling for your AIN input the input is scaled down to 1/3. The effect is that your AIN voltage range is 0 - 3.6 V because the 3.6 V input voltage is scaled down to 3.6 / 3 = 1.2 V. *Table 311: Saturation point examples* on page 165 shows the ADC range for different reference voltages and prescalers.

### Table 311: Saturation point examples

| Reference | Prescaling | ADC range |
|---|---|---|
| 1.2 V VBG | 1 | 0 - 1.2 V |
| 1.2 V VBG | 2/3 | 0 - 1.8 V |
| 1.2 V VBG | 1/3 | 0 - 3.6 V |
| 1.0 V AREF | 1 | 0 - 1.0 V |
| 1.0 V AREF | 2/3 | 0 - 1.5 V |
| 1.0 V AREF | 1/3 | 0 - 3.0 V |
| VDD 3.0 V, VDD 1/2 | 1 | 0 - 1.5 V |

#### Voltage divider

There are two rules to follow to find the maximum input voltage allowed on the AIN pins:

- **Rule 1:** The ADC should not be exposed to higher voltage than 2.4 V on an AIN pin with no prescaling setting (prescaling = 1).
- **Rule 2:** A GPIO pin must not be exposed to higher voltage than VDD + 0.3 V, according to the Absolute maximum ratings in the nRF51x22 Product Specification.

For example, when using 2/3 prescaling, you can expose 2.4 V / (2/3) = 3.6 V to an AIN pin. To not violate rule 2, VDD must be 3.3 V or higher.

Table 2 shows examples on maximum voltages that can be exposed to an ADC AIN pin, depending on the supply voltage and your prescaling settings

### Table 312: AIN maximum voltage examples

| Supply voltage | Prescaling | AIN max. voltage | Rule limitation |
|---|---|---|---|
| 3.6 V | 1 | 2.4 V | Rule 1 |
| 3.6 V | 2/3 | 3.6 V | Rule 1 |
| 3.6 V | 1/3 | 3.9 V | Rule 2 |
| 3.3 V | 1 | 2.4 V | Rule 1 |
| 3.3 V | 2/3 | 3.6 V | Rule 1 and Rule 2 |
| 3.3 V | 1/3 | 3.6 V | Rule 2 |
| 1.8 V | 1 | 2.1 V | Rule 2 |
| 1.8 V | 2/3 | 2.1 V | Rule 2 |
| 1.8 V | 1/3 | 2.1 V | Rule 2 |

If the signal you want to measure is above the maximum allowed AIN voltage, a voltage divider must be used. See *Using a voltage divider to lower ADC input voltage* on page 165.

## 31.1.5 Using a voltage divider to lower ADC input voltage

If a sensor or battery has output voltage above the ADC voltage range, it is necessary to lower that voltage before exposing it to an ADC input pin. This can be achieved with a voltage divider. *Figure 72: Voltage divider on ADC input* on page 166 shows how a voltage divider can be connected to the ADC to lower the input voltage.

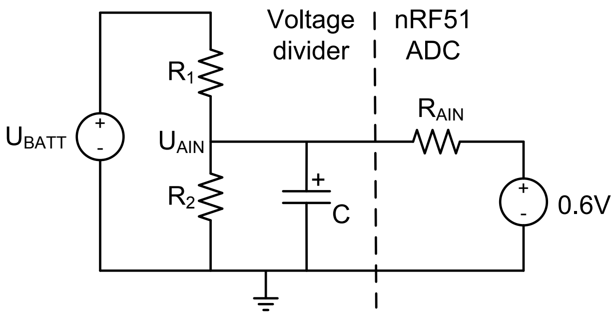



**Figure 72: Voltage divider on ADC input**

Because of internal impedance of the ADC, having a voltage divider with large resistor values will introduce error in ADC output. If the impedance of the voltage divider is less than 1 kΩ, the error is very small and can be neglected. As the impedance of the voltage divider is increased, the error will also increase.

But it can also be desirable to have high resistor values in the voltage divider to limit the current leak through the voltage divider. A way to reduce the error introduced by the high resistance values is to add a capacitor between the AIN pin and ground. The higher the capacitor value is, the more it will decrease the ADC output error, but it will also reduce the sampling frequency accordingly.

The moment you are sampling, $R_{AIN}$ is 120 - 400 kΩ (see *Table 313: Input impedance statistics for RAIN* on page 166) and therefore lowers the $U_{AIN}$ voltage when a voltage divider is connected. If a capacitor also is connected between AIN and ground, it will keep the $U_{AIN}$ voltage at the previous level for an adequate time period while sampling, therefore minimizing the effect of the high resistance value of $R_2$. The capacitor must be large enough to hold the voltage up for the required time period, i.e. 20 µs for 8-bit sampling or 68 µs for 10-bit sampling. The capacitor must also be small enough to fully charge before the next sample is taken. So when a capacitor is connected, it's size is a trade-off between accuracy and sampling frequency. When not sampling, the $R_{AIN}$ will have very high value and you can consider it to be an open circuit.

For input voltages above the ADC voltage range and where high accuracy and high sampling frequency is needed, a voltage buffer is needed.

Another possible method is to connect a FET transistor between the power supply and the voltage divider which will open for current through the voltage divider momentarily before sampling. The voltage divider can then have low resistor values (<1 kΩ) and no capacitor is needed. The voltage divider would then consume relatively high current when sampling, but will not consume any current when not sampling.

## 31.1.6 Input impedance

To achieve the ADC error specifications stated in the nRF51x22 Product Specification, the output impedance of the connected voltage source must be 1 kΩ or lower. Another advantage if the output impedance is 1 kΩ or lower is that different prescaling settings for the ADC input will have no practical effect on the ADC accuracy.

If a voltage source with higher impedance is applied, additional gain and offset error is introduced, which also will vary for different prescaler settings.

*Figure 72: Voltage divider on ADC input* on page 166 shows the nRF51 ADC input model when the ADC is sampling and *Table 313: Input impedance statistics for RAIN* on page 166 shows the value of $R_{AIN}$ for different prescaling settings. The internal VBG reference voltage is 1.2 V so the ADC internal voltage source is VBG/2 = 0.6 V.

When the ADC is not sampling the AIN input pin has very high impedance and can be regarded as open circuit. *Table 313: Input impedance statistics for RAIN* on page 166 shows the statistics for the internal impedance for different prescaling settings. 99.7% of devices (+- 3 sigma) are expected to be within 6.3%, for example for 1/1 prescaling => [121.5, 137.9] kΩ.

**Table 313: Input impedance statistics for $R_{AIN}$**

| Prescaling | Mean impedance | Standard deviation |
|---|---|---|
| 1/1 | 129.7 kΩ | 2.74 kΩ |
| 2/3 | 194.6 kΩ | 4.1 kΩ |
| 1/3 | 389.2 kΩ | 8.2 kΩ |

### 31.1.7 Shared resources

The ADC is using the same analog pins as the LPCOMP. The ADC is using the same analog input pins as the LPCOMP and they also share the MUX shown in *Figure 71: Analog to digital converter* on page 164. The ADC and LPCOMP should therefore not have any analog input pins configured at the same time, i.e. only one of them should configure analog input pins.

## 31.2 Register Overview

### Table 314: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40007000 | ADC | ADC | Analog to Digital Converter |

### Table 315: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *START* | 0x000 | Start a new ADC conversion |
| *STOP* | 0x004 | Stop ADC |
| Events | | |
| *END* | 0x100 | An ADC conversion is completed |
| Registers | | |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *BUSY* | 0x400 | ADC busy (conversion in progress) |
| *ENABLE* | 0x500 | Enable ADC. When enabled, the ADC will acquire access to the analog input pins specified in the CONFIG register. |
| *CONFIG* | 0x504 | ADC configuration |
| *RESULT* | 0x508 | Result of the previous ADC conversion |

## 31.3 Register Details

### Table 316: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END | | | Enable or disable interrupt on *END* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

### Table 317: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END | | | Write '1' to Enable interrupt on *END* event. |
| | | | Enabled | 1 | Enable |

### Table 318: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | END | | | Write '1' to Clear interrupt on *END* event. |
| | | | Disabled | 1 | Disable |

### Table 319: BUSY

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | BUSY | | | ADC busy 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| | | | Ready | 0 | ADC is ready. No ongoing conversion. |
| | | | Busy | 1 | ADC is busy. Conversion in progress. |

## Table 320: 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 | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | ADC enable |
| | | | Disabled | 0 | ADC disabled |
| | | | Enabled | 1 | ADC enabled |

## Table 321: CONFIG

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | D | D | D | D | D | | C | C | B | B | B | A | A |
| Reset | 0 | 0 | 0 | 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 | 0 | 0 |

| Id | RW | Field | Value Id | Value | Description |
|---|---|---|---|---|---|
| A | RW | RES | | | ADC resolution |
| | | | 8bit | 0 | 8 bit |
| | | | 9bit | 1 | 9 bit |
| | | | 10bit | 2 | 10 bit |
| B | RW | INPSEL | | | ADC input selection |
| | | | AnalogInputNoPrescaling | 0 | Analog input pin specified by CONFIG.PSEL with no prescaling |
| | | | AnalogInputTwoThirdsPre: | 1 | Analog input pin specified by CONFIG.PSEL with 2/3 prescaling |
| | | | AnalogInputOneThirdPres( | 2 | Analog input pin specified by CONFIG.PSEL with 1/3 prescaling |
| | | | SupplyTwoThirdsPrescalin | 5 | VDD with 2/3 prescaling |
| | | | SupplyOneThirdPrescaling | 6 | VDD with 1/3 prescaling |
| C | RW | REFSEL | | | ADC reference selection |
| | | | VBG | 0 | Use internal 1.2 V band gap reference |
| | | | External | 1 | Use external reference specified by CONFIG. EXTREFSEL |
| | | | SupplyOneHalfPrescaling | 2 | Use VDD with 1/2 prescaling. (Only applicable when VDD is in the range 1.7 V - 2.6 V). |
| | | | SupplyOneThirdPrescaling | 3 | Use VDD with 1/3 prescaling. (Only applicable when VDD is in the range 2.5 V - 3.6 V). |
| D | RW | PSEL | | | Select pin to be used as ADC input pin |
| | | | Disabled | 0 | Analog input pins disabled |
| | | | AnalogInput0 | 1 | Use AIN0 as analog input |
| | | | AnalogInput1 | 2 | Use AIN1 as analog input |
| | | | AnalogInput2 | 4 | Use AIN2 as analog input |
| | | | AnalogInput3 | 8 | Use AIN3 as analog input |
| | | | AnalogInput4 | 16 | Use AIN4 as analog input |
| | | | AnalogInput5 | 32 | Use AIN5 as analog input |
| | | | AnalogInput6 | 64 | Use AIN6 as analog input |
| | | | AnalogInput7 | 128 | Use AIN7 as analog input |
| E | RW | EXTREFSEL | | | External reference pin selection |
| | | | None | 0 | Analog reference inputs disabled |
| | | | AnalogReference0 | 1 | Use AREF0 as analog reference |
| | | | AnalogReference1 | 2 | Use AREF1 as analog reference |

## Table 322: RESULT

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RESULT | | [0..1023] | Result of the previous ADC conversion<br>The value is updated for every completed ADC conversion. The result value is relative to the selected ADC reference input.<br>If the sampled analog input signal is equal to or greater than the ADC reference signal, the result value will be set to the maximum (limited by the selected ADC bit width). The value is right justified (LSB of sample value always on register bit 0). |

# 32 Low Power Comparator (LPCOMP)

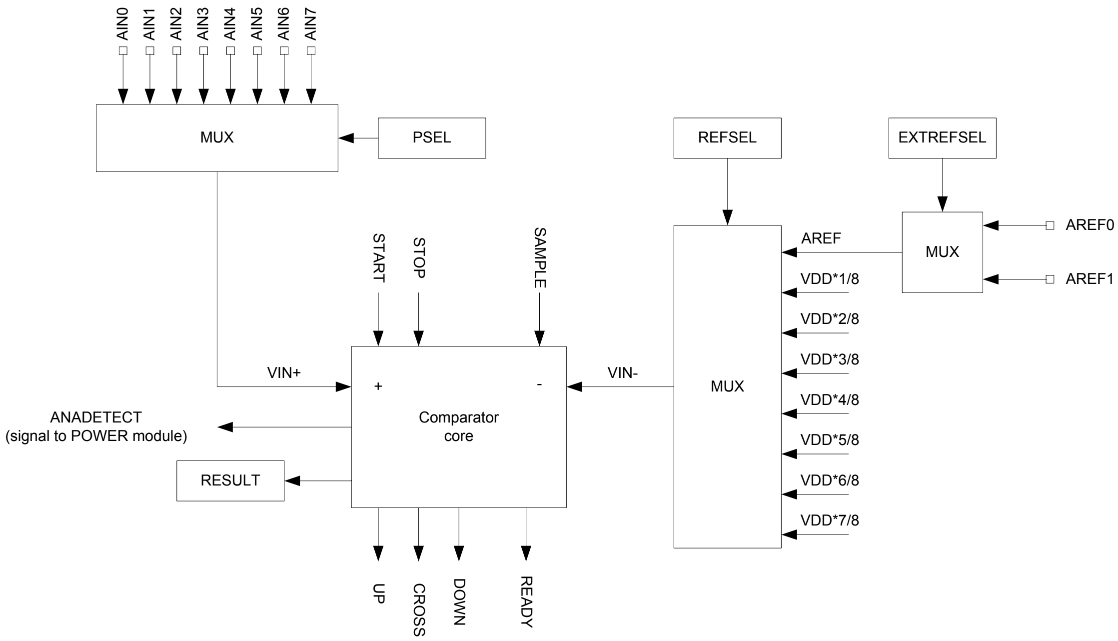



**Figure 73: Low power comparator**

## 32.1 Functional description

The low power comparator (LPCOMP) compares an input voltage (VIN+), which comes from an analog input pin selected through the PSEL register against a reference voltage (VIN-) selected through the REFSEL and EXTREFSEL registers.

The PSEL, REFSEL, and EXTREFSEL registers must be configured before the LPCOMP is enabled through the ENABLE register.

Specific chip variants may not offer all the reference and/or analog inputs defined here.

The LPCOMP is started by triggering the START task. After a start-up time of $t_{LPCOMPSTARTUP}$ the LPCOMP will generate a READY event to indicate that the comparator is ready to use and the output of the LPCOMP is correct. The LPCOMP will generate events every time VIN+ crosses VIN-. More specifically, every time VIN+ rises above VIN- (upward crossing) an UP event is generated along with a CROSS event. Every time VIN+ falls below VIN- (downward crossing), a DOWN event is generated along with a CROSS event.

The LPCOMP is stopped by triggering the STOP task.

LPCOMP will be operational in both System ON and System OFF mode when it is enabled through the ENABLE register, see *Power management (POWER)* on page 42 for more information about power modes. All LPCOMP registers including the ENABLE register are classified as retained registers when the LPCOMP is enabled. However, when the device wakes up from System OFF, all LPCOMP registers will be reset.

The LPCOMP can wake up the system from System OFF by asserting the ANADETECT signal. The ANADETECT signal can be derived from any of the event sources that generate the UP, DOWN and CROSS events. In case of wakeup from System OFF, no events will be generated, only the ANADETECT signal. See the ANADETECT register (*Table 334: ANADETECT* on page 172) for more information on how to configure the ANADETECT signal.

The immediate value of the LPCOMP can be sampled to the RESULT register by triggering the SAMPLE task.

See the RESETREAS register in the POWER module (*Table 53: RESETREAS* on page 48) for more information on how to detect a wakeup from LPCOMP.

## 32.2 Pin configuration

You can use the PSEL register to select one of the analog input pins, AIN0 through AIN7, as analog input pin for the LPCOMP, see *Figure 73: Low power comparator* on page 169. Similarly, you can use the EXTREFSEL register to select one of the analog reference input pins, AREF0 and AREF1, as input for AREF in case AREF is selected in REFSEL. The selected analog pins will be acquired by the LPCOMP when it is enabled through the ENABLE register. See the product specification for more information about which analog pins are available on a particular device.

## 32.3 Shared resources

The LPCOMP shares registers and other resources with peripherals that have the same ID as the LPCOMP. You must disable all peripherals that have the same ID as the LPCOMP before the LPCOMP can be configured and used. Disabling a peripheral that has the same ID as the LPCOMP will not reset any of the registers that are shared with the LPCOMP. Therefore, it is important to configure all relevant LPCOMP registers explicitly to secure that it operates correctly.

See the Instantiation table in *Instantiation* on page 17 for details on peripherals and their IDs.

> **Note:** The LPCOMP is using the same analog pins as the ADC. The ADC must be disabled before the LPCOMP can be enabled.

## 32.4 Register Overview

### Table 323: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40013000 | LPCOMP | LPCOMP | Low Power Comparator |

### Table 324: Register Overview

| Register | Offset | Description |
|---|---|---|
| Tasks | | |
| *START* | 0x000 | Start comparator |
| *STOP* | 0x004 | Stop comparator |
| *SAMPLE* | 0x008 | Sample comparator value |
| Events | | |
| *READY* | 0x100 | LPCOMP is ready and output is valid |
| *DOWN* | 0x104 | Downward crossing |
| *UP* | 0x108 | Upward crossing |
| *CROSS* | 0x10C | Downward or upward crossing |
| Registers | | |
| *SHORTS* | 0x200 | Shortcut register |
| *INTEN* | 0x300 | Enable or disable interrupt |
| *INTENSET* | 0x304 | Enable interrupt |
| *INTENCLR* | 0x308 | Disable interrupt |
| *RESULT* | 0x400 | Compare result |
| *ENABLE* | 0x500 | Enable LPCOMP |
| *PSEL* | 0x504 | Input pin select |
| *REFSEL* | 0x508 | Reference select |
| *EXTREFSEL* | 0x50C | External reference select |
| *ANADETECT* | 0x520 | Analog detect configuration |

## 32.5 Register Details

### Table 325: SHORTS

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY_SAMPLE | | | Shortcut between *READY* event and *SAMPLE* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| B | RW | READY_STOP | | | Shortcut between *READY* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |
|---|---|---|---|---|---|
| C | RW | DOWN_STOP | | | Shortcut between *DOWN* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| D | RW | UP_STOP | | | Shortcut between *UP* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |
| E | RW | CROSS_STOP | | | Shortcut between *CROSS* event and *STOP* task |
| | | | Disabled | 0 | Disable shortcut |
| | | | Enabled | 1 | Enable shortcut |

## Table 326: INTEN

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY | | | Enable or disable interrupt on *READY* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| B | RW | DOWN | | | Enable or disable interrupt on *DOWN* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| C | RW | UP | | | Enable or disable interrupt on *UP* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |
| D | RW | CROSS | | | Enable or disable interrupt on *CROSS* event |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

## Table 327: INTENSET

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY | | | Write '1' to Enable interrupt on *READY* event. |
| | | | Enabled | 1 | Enable |
| B | RW | DOWN | | | Write '1' to Enable interrupt on *DOWN* event. |
| | | | Enabled | 1 | Enable |
| C | RW | UP | | | Write '1' to Enable interrupt on *UP* event. |
| | | | Enabled | 1 | Enable |
| D | RW | CROSS | | | Write '1' to Enable interrupt on *CROSS* event. |
| | | | Enabled | 1 | Enable |

## Table 328: INTENCLR

**Note:** Write '0' has no effect. When read this register will return the value of *INTEN*.

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | READY | | | Write '1' to Clear interrupt on *READY* event. |
| | | | Disabled | 1 | Disable |
| B | RW | DOWN | | | Write '1' to Clear interrupt on *DOWN* event. |
| | | | Disabled | 1 | Disable |
| C | RW | UP | | | Write '1' to Clear interrupt on *UP* event. |
| | | | Disabled | 1 | Disable |
| D | RW | CROSS | | | Write '1' to Clear interrupt on *CROSS* event. |
| | | | Disabled | 1 | Disable |

## Table 329: RESULT

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | R | RESULT | | | Result of last compare. Decision point SAMPLE task. |
| | | | Bellow | 0 | Input voltage is below the reference threshold (VIN+ < VIN-). |
| | | | Above | 1 | Input voltage is above the reference threshold (VIN+ > VIN-). |

## Table 330: 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 | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ENABLE | | | Enable or disable LPCOMP |
| | | | Disabled | 0 | Disable |
| | | | Enabled | 1 | Enable |

## Table 331: 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 | A | A |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | PSEL | | | Analog pin select |
| | | | AnalogInput0 | 0 | AIN0 selected as analog input |
| | | | AnalogInput1 | 1 | AIN1 selected as analog input |
| | | | AnalogInput2 | 2 | AIN2 selected as analog input |
| | | | AnalogInput3 | 3 | AIN3 selected as analog input |
| | | | AnalogInput4 | 4 | AIN4 selected as analog input |
| | | | AnalogInput5 | 5 | AIN5 selected as analog input |
| | | | AnalogInput6 | 6 | AIN6 selected as analog input |
| | | | AnalogInput7 | 7 | AIN7 selected as analog input |

## Table 332: REFSEL

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | REFSEL | | | Reference select |
| | | | SupplyOneEighthPrescalin | 0 | VDD * 1/8 selected as reference |
| | | | SupplyTwoEighthsPrescali | 1 | VDD * 2/8 selected as reference |
| | | | SupplyThreeEighthsPresca | 2 | VDD * 3/8 selected as reference |
| | | | SupplyFourEighthsPrescali | 3 | VDD * 4/8 selected as reference |
| | | | SupplyFiveEighthsPrescali | 4 | VDD * 5/8 selected as reference |
| | | | SupplySixEighthsPrescalin | 5 | VDD * 6/8 selected as reference |
| | | | SupplySevenEighthsPresca | 6 | VDD * 7/8 selected as reference |
| | | | ARef | 7 | External analog reference selected |

## Table 333: EXTREFSEL

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | EXTREFSEL | | | External analog reference select |
| | | | AnalogReference0 | 0 | Use AREF0 as external analog reference |
| | | | AnalogReference1 | 1 | Use AREF1 as external analog reference |

## Table 334: ANADETECT

| Bit number | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | ANADETECT | | | Analog detect configuration |
| | | | Cross | 0 | Generate ANADETECT on crossing, both upward crossing and downward crossing |
| | | | Up | 1 | Generate ANADETECT on upward crossing only |
| | | | Down | 2 | Generate ANADETECT on downward crossing only |

# 33 Software Interrupts (SWI)

## 33.1 Functional description

A set of interrupts have been reserved for use as software interrupts.

## 33.2 Register Overview

### Table 335: Instances

| Base address | Peripheral | Instance | Description |
|---|---|---|---|
| 0x40014000 | SWI | SWI0 | Software interrupt |
| 0x40015000 | SWI | SWI1 | Software interrupt |
| 0x40016000 | SWI | SWI2 | Software interrupt |
| 0x40017000 | SWI | SWI3 | Software interrupt |
| 0x40018000 | SWI | SWI4 | Software interrupt |
| 0x40019000 | SWI | SWI5 | Software interrupt |