EFR32xG22 Wireless Gecko Reference Manual

The EFR32xG22 Wireless Gecko SoC includes the EFR32BG22, EFR32FG22, and EFR32MG22 Wireless Gecko families. The EFR32xG22 improves processing capability with a Cortex M33 core, while providing for lower active current for both the MCU and radio. This low power and application optimized device supports Bluetooth 5.2 (including Direction Finding), Proprietary 2.4 GHz & Zigbee PRO/Green Power protocols.

The EFR32xG22 solution provides industry-leading energy efficiency, processing capability, and RF performance in a small form factor for IoT connected applications.

KEY FEATURES

- 32-bit ARM® Cortex M33 core with up to 76.8 MHz maximum operating frequency
- Scalable Memory and Radio configuration options available in QFN packaging
- Energy-efficient radio core with low active and sleep currents
- Secure Boot with Root of Trust and Secure Loader (RTSL)
- Integrated PA with up to 6 dBm transmit power

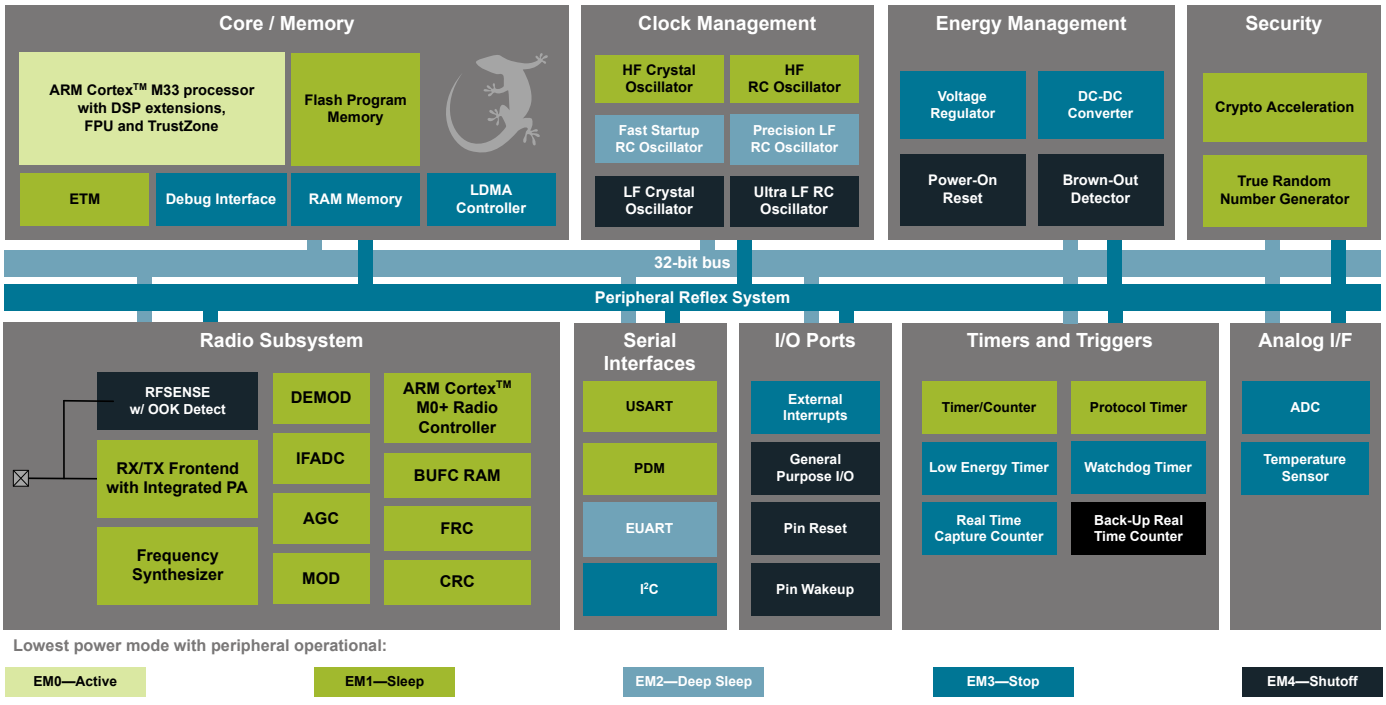


Table of Contents

1. About This Document

1.1 Introduction

This document contains reference material for the EFR32xG22 devices. All modules and peripherals in the EFR32xG22 devices are described in general terms. Not all modules are present in all devices and the feature set for each device might vary. Such differences, including pinout, are covered in the device data sheets.

1.2 Conventions

Register Names

Register names are given with a module name prefix followed by the short register name:

TIMERn_CTRL - Control Register

The "n" denotes the module number for modules which can exist in more than one instance.

Some registers are grouped which leads to a group name following the module prefix:

GPIO_Px_DOUT - Port Data Out Register

The "x" denotes the different ports.

Bit Fields

Registers contain one or more bit fields which can be 1 to 32 bits wide. Bit fields wider than 1 bit are given with start (x) and stop (y) bit [y:x].

Bit fields containing more than one bit are unsigned integers unless otherwise is specified.

Unspecified bit field settings must not be used, as this may lead to unpredictable behaviour.

Address

The address for each register can be found by adding the base address of the module found in the Memory Map (see Figure 4.1 System Address Space with Core and Code Space Listing on page 37), and the offset address for the register (found in module Register Map).

Access Type

The register access types used in the register descriptions are explained in Table 1.1 Register Access Types on page 25.

Table 1.1. Register Access Types

Access Type	Description
R	Read only. Writes are ignored
RW	Readable and writable
RW1	Readable and writable. Only writes to 1 have effect
(R)W1	Sometimes readable. Only writes to 1 have effect. Currently only used for IF_CLEAR registers (see 3.3.1 Interrupt Operation)
W1	Read value undefined. Only writes to 1 have effect
W	Write only. Read value undefined.
RWH	Readable, writable, and updated by hardware
RW(nB), RWH(nB), etc.	"(nB)" suffix indicates that register explicitly does not support peripheral bit set or clear (see 4. Memory and Bus System)
RW(a), R(a), etc.	"(a)" suffix indicates that reading the register cause an action and ay alter the register value.

Number format

0x prefix is used for hexadecimal numbers

0b prefix is used for binary numbers

Numbers without prefix are in decimal representation.

Reserved

Registers and bit fields marked with **reserved** are reserved for future use. These should be written to 0 unless otherwise stated in the Register Description. Reserved bits might be read as 1 in future devices.

Reset Value

The reset value denotes the value after reset.

Registers denoted with X have unknown value out of reset and need to be initialized before use. Note that read-modify-write operations on these registers before they are initialized results in undefined register values.

Pin Connections

Pin connections are given with a module prefix followed by a short pin name:

CMU_CLKOUT1 (Clock management unit, clock output pin number 1)

The location for the pin names given in the module documentation can be found in the device-specific datasheet.

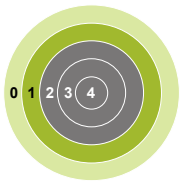
1.3 Related Documentation

Further documentation on the EFR32xG22 devices and the ARM Cortex-M33 can be found at the Silicon Labs and ARM web pages:

www.silabs.com

www.arm.com

2. System Overview



Quick Facts

What?

The EFR32 Wireless Gecko is a highly integrated, configurable and low power wireless System-on-Chip (SoC) with a robust set of MCU and radio peripherals.

Why?

The Radio enables support for Bluetooth, Proprietary, and ZigBee protocols in 2.4 GHz frequency bands while the MCU system allows customized protocols and applications to run efficiently.

How?

Dynamic or fixed packet lengths, optional address recognition, and flexible CRC and security schemes makes the EFR32xG22 ideal for many wireless IoT applications. High performance analog and digital peripherals allow complete applications to run on the EFR32xG22 SoC.

2.1 Introduction

The high level features of EFR32xG22 include:

- High performance radio transceiver
  - Low power consumption in transmit, receive, and standby modes
  - Excellent receiver performance, including sensitivity, selectivity, and blocking
  - Excellent transmitter performance, including programmable output power, low phase noise, and power-amplifier (PA) ramping
  - Ultra-low energy RF detection for wake-up from any energy mode, through RFSENSE
- Configurable protocol support, including standards and customer-developed protocols
  - Preamble and frame synchronization insertion in transmit, and recovery in receive
  - Flexible CRC support, including configurable polynomial and multiple CRCs for single data frames
  - Basic address filtering performed in hardware
- High performance, low power MCU system
  - High Performance 32-bit ARM Cortex-M33 CPU
  - Flexible and efficient energy management
  - Complete set of digital peripherals
  - Peripheral Reflex System (PRS)
  - Precision analog peripherals
- Low external component count
  - Fully integrated 2.4 GHz BALUN
  - Integrated tunable crystal loading capacitors

A further introduction to the MCU and radio system is included in the following sections.

Note: Detailed performance numbers, current consumption, pinout etc. are available in the device datasheets.

2.2 Block Diagrams

The block diagram for the EFR32xG22 System-On-Chip series is shown in (Figure 2.1 EFR32xG22 System-On-Chip Block Diagram on page 28).

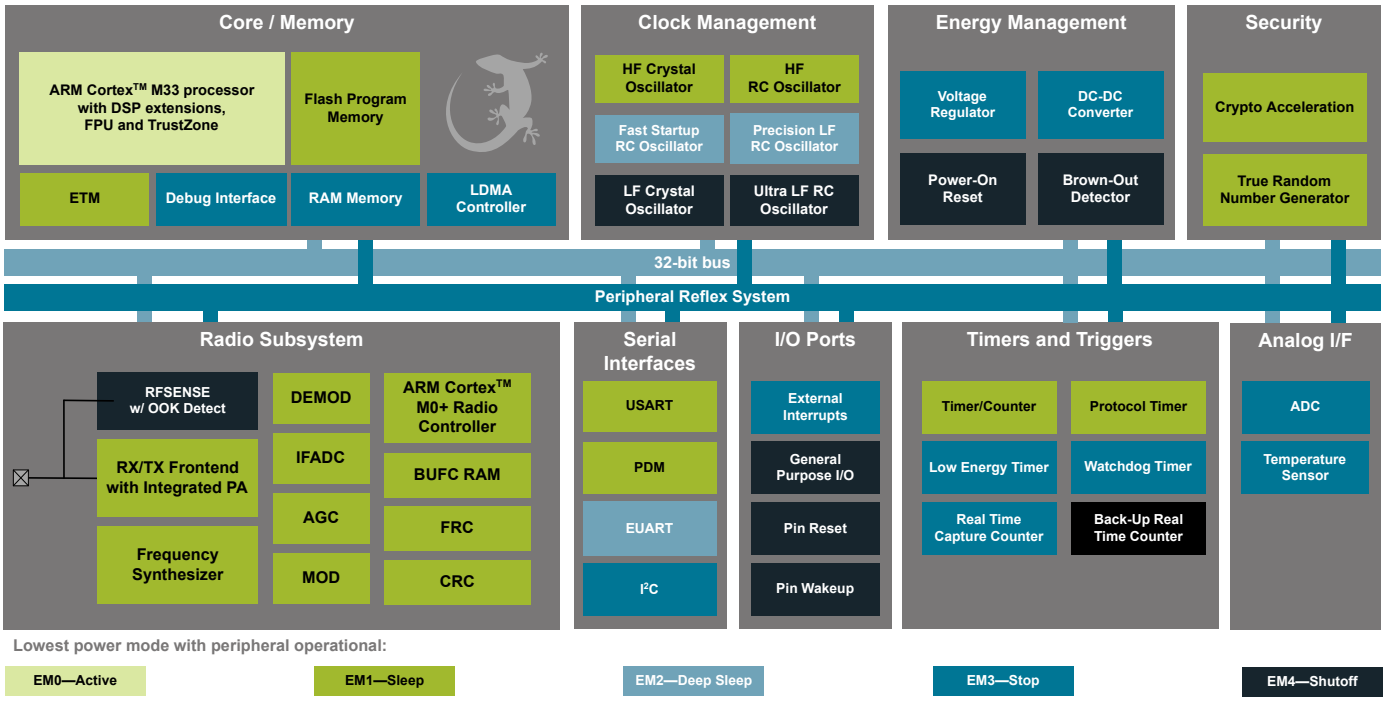


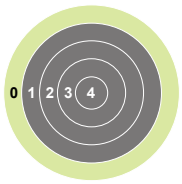
Figure 2.1. EFR32xG22 System-On-Chip Block Diagram

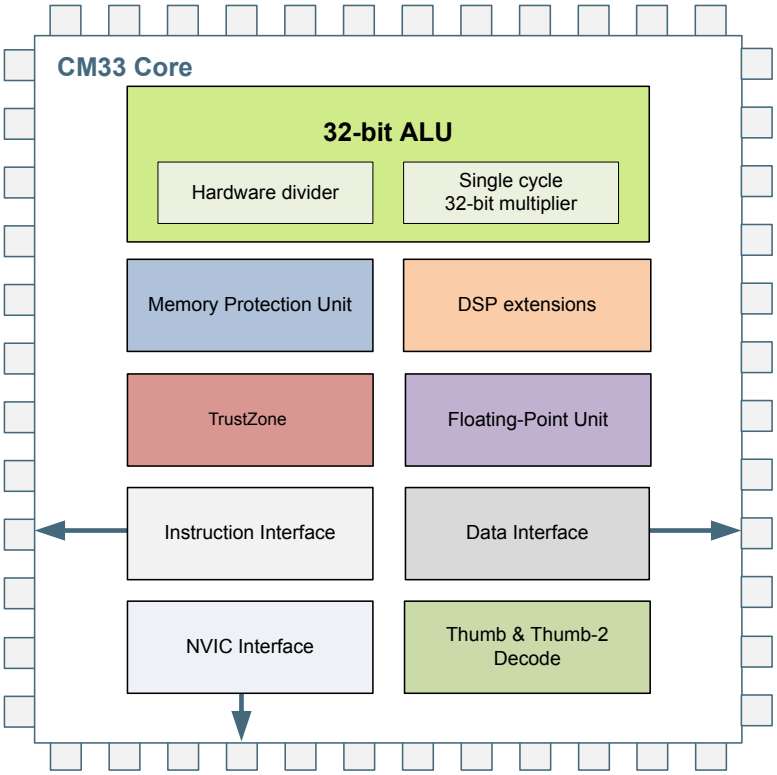
2.3 MCU Features overview

- **ARM Cortex-M33 CPU platform**
  - High Performance 32-bit processor @ up to 76.8 MHz
  - DSP instruction support and floating-point unit
  - Memory Protection Unit
  - Wake-up Interrupt Controller
- **Flexible Energy Management System**
  - Five Energy Modes from EM0 to EM4 provide flexibility between higher performance and low power
  - Power routing configurations including DCDC control
  - Voltage Monitoring and Brown Out Detection
  - Automatic voltage scaling for additional energy savings
  - State Retention
  - Selective RFSENSE wakeup source
- **Up to 512 kB Flash**
- **Up to 32 kB RAM**
- **Up to 26 General Purpose I/O pins**
  - Configurable push-pull, open-drain, pull-up/down, input filter, slew rate
  - Configurable peripheral I/O locations
  - 16 asynchronous external interrupts
  - Output state retention and wake-up from Shutoff Mode
- **8 Channel DMA Controller**
  - Alternate/primary descriptors with scatter-gather/ping-pong operation
- **16 Channel Peripheral Reflex System (PRS)**
  - Autonomous inter-peripheral signaling enables smart operation in low energy modes
  - 12 asynchronous channels with configurable logic functionality
  - 4 synchronous channels for high-speed signalling between TIMER and IADC
- **Cryptographic Accelerator (CRYPTOACC)**
  - AES encryption / decryption, with 128 or 256 bit keys
  - Multiple AES modes of operation, including Counter (CTR), Electronic CodeBook (ECB), Cipher Block Chaining (CBC), Counter mode with CBC-MAC (CCM), and Cipher-based Message Authentication Code (CMAC).
  - Accelerated SHA-1 and SHA-2 (SHA-224 / SHA-256)
  - Accelerated Elliptic Curve Cryptography (ECC), with binary or prime fields
  - Flexible 256-bit ALU and sequencer
  - True random number generation
- **General Purpose Cyclic Redundancy Check (GPCRC)**
  - Programmable 16-bit polynomial, fixed 32-bit polynomial
  - The GPCRC module is in addition to the radio CRC
- **Communication interfaces**
  - 2 × Universal Synchronous/Asynchronous Receiver/Transmitter (USART)
    - UART/SPI/SmartCard (ISO 7816)/IrDA/I2S
    - Triple buffered full/half-duplex operation
    - Hardware flow control
    - 4-16 data bits
  - Enhanced Universal Asynchronous Receiver/Transmitter (EUART)
    - UART and IrDA support
    - High-speed operation in EM0/1 using high-frequency clock source
    - Low-energy operation in EM2 using 32.768 kHz clock source
    - Buffered full/half-duplex operation
    - Hardware flow control
    - 7/8/9 data bits
  - 2 × I^2C Interface (I2C) with SMBus support
    - Address recognition in EM3 Stop Mode

- **Timers/Counters**
  - 1 × 32-bit and 4 × 16-bit Timer/Counters (TIMER)
    - 3 Compare/Capture/PWM channels
    - Dead-Time Insertion
  - 24-bit Low Energy Timer (LETIMER)
  - 32-bit Real-Time Capture Counter (RTCC)
  - 32-bit Ultra Low Energy Backup Real Time Counter (BURTC) for periodic wake-up from any Energy Mode
  - Watchdog Timer (WDOG)
- **Ultra low power precision analog peripherals**
  - Incremental Analog to Digital Converter (IADC) with 12-bit resolution at 1 Msps and 16-bit resolution at 76.9 ksps
    - Single ended or differential operation
    - Conversion tailgating for predictable latency
  - Analog Bus (ABUS) signal routing
  - Accurate die temperature sensor
  - External thermistor interface
- **Ultra efficient Power-on Reset (POR) and Brown-Out Detector (BOD)**
- **Debug Interface**
  - 4-pin Joint Test Action Group (JTAG) interface
  - 2-pin serial-wire debug (SWD) interface

3. System Processor





Quick Facts

What?

The EFR32xG22 features the industry leading Cortex-M33 CPU from ARM.

Why?

The ARM Cortex-M33 is designed for exceptionally short response time, high code density, and high 32-bit throughput while maintaining a strict cost and power consumption budget.

How?

Combined with the ultra low energy peripherals available in EFR32xG22 devices, the Cortex-M33 processor's Harvard architecture, 3 stage pipeline, single cycle instructions, Thumb-2 instruction set support, and fast interrupt handling make it perfect for 8-bit, 16-bit, and 32-bit applications.

3.1 Introduction

The ARM Cortex-M33 32-bit RISC processor provides outstanding computational performance and exceptional system response to interrupts while meeting low cost requirements and low power consumption.

The ARM Cortex-M33 implemented is revision r0p4.

3.2 Features

- Harvard architecture
  - Separate data and program memory buses (No memory bottleneck as in a single bus system)
- 3-stage pipeline
- Thumb-2 instruction set
  - Enhanced levels of performance, energy efficiency, and code density
- Single cycle multiply and hardware divide instructions
  - 32-bit multiplication in a single cycle
  - Signed and unsigned divide operations between 2 and 11 cycles
- 1.5 DMIPS/MHz
- TrustZone
  - Independent Secure and Privileged states
  - Accelerated context switching
- 16 Region MPU
- 24-bit System Tick Timer for Real Time OS
- Excellent 32-bit migration choice for 8/16 bit architecture based designs
  - Simplified stack-based programmer's model is compatible with traditional ARM architecture and retains the programming simplicity of legacy 8-bit and 16-bit architectures
- Aligned or unaligned data storage and access
  - Contiguous storage of data requiring different byte lengths
  - Data access in a single core access cycle
- Integrated power modes
  - Sleep Now mode for immediate transfer to low power state
  - Sleep on Exit mode for entry into low power state after the servicing of an interrupt
  - Ability to extend power savings to other system components
- Optimized for low latency, nested interrupts

3.3 Functional Description

For a full functional description of the ARM Cortex-M33 implementation in the EFR32xG22 family, the reader is referred to the ARM Cortex-M33 documentation.

3.3.1 Interrupt Operation

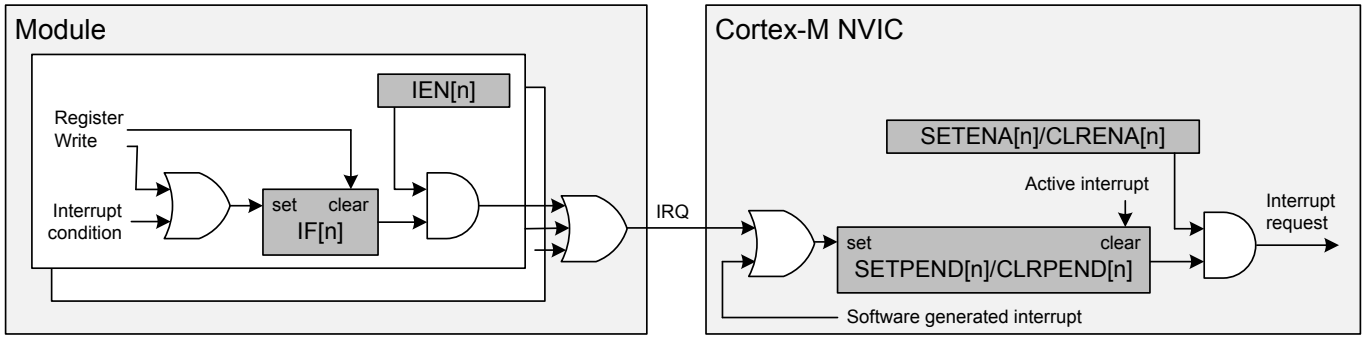


Figure 3.1. Interrupt Operation

The interrupt request (IRQ) lines are connected to the Cortex-M33. Each of these lines (shown in 3.3.3 Interrupt Request lines (IRQ)) is connected to one or more interrupt flags in one or more modules. The interrupt flags are set by hardware on an interrupt condition. It is also possible to set/clear the interrupt flags through the IF registers. When setting or clearing and interrupt through the IF register use of the IF_SET or IF_CLEAR bit operation registers is recommended.

Each interrupt flag is then qualified with its own interrupt enable bit (IEN register), before being OR'ed with the other interrupt flags to generate the IRQ. A high IRQ line will set the corresponding pending bit (can also be set/cleared with the SETPEND/CLRPEND bits in ISPRn/ICPRn) in the Cortex-M33 NVIC. The pending bit is then qualified with an enable bit (set/cleared with SETENA/CLRENA bits in ISERn/ICERn) before generating an interrupt request to the core. Figure 3.1 Interrupt Operation on page 33 illustrates the interrupt system. For more information on how the interrupts are handled inside the Cortex-M33, the reader is referred to the **ARM Cortex-M33 Processor Technical Reference Manual**.

3.3.1.1 Avoiding Extraneous Interrupts

There can be latencies in the system such that clearing an interrupt flag could take longer than leaving an Interrupt Service Routine (ISR). This can lead to the ISR being re-entered as the interrupt flag has yet to clear immediately after leaving the ISR. To avoid this, when clearing an interrupt flag at the end of an ISR, the user should execute ARM's Data Synchronization Barrier (DSB) instruction. Another approach is to clear the interrupt flag immediately after identifying the interrupt source and then service the interrupt as shown in the pseudo-code below. The ISR typically is sufficiently long to more than cover the few cycles it may take to clear the interrupt status, and also allows the status to be checked for further interrupts before exiting the ISR.

```
irqXServiceRoutine() {
   do {
      clearIrqXStatus();
      serviceIrqX();
   } while(irqXStatusIsActive());
}
```

3.3.2 TrustZone

The Cortex-M33 implements ARM TrustZone which provides the ability to restrict access to peripherals and memory regions based on the CPU security attribute. TrustZone works in combination which the MPU which controls privileged/unprivileged execution of code to provide a full security solution. The Security Management Unit (SMU) is used to configure access restrictions in the various modes. Refer to 10. SMU - Security Management Unit for more information.

For information about TrustZone features in the core or information on TrustZone specific instructions please see the ARM Cortex-M33 Processor Technical Reference Manual provided by ARM

3.3.3 Interrupt Request lines (IRQ)

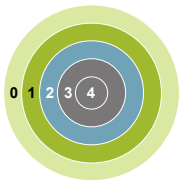
This table shows all IRQ's for the system processor. M33 High Speed interrupts are indicated by an '*'.

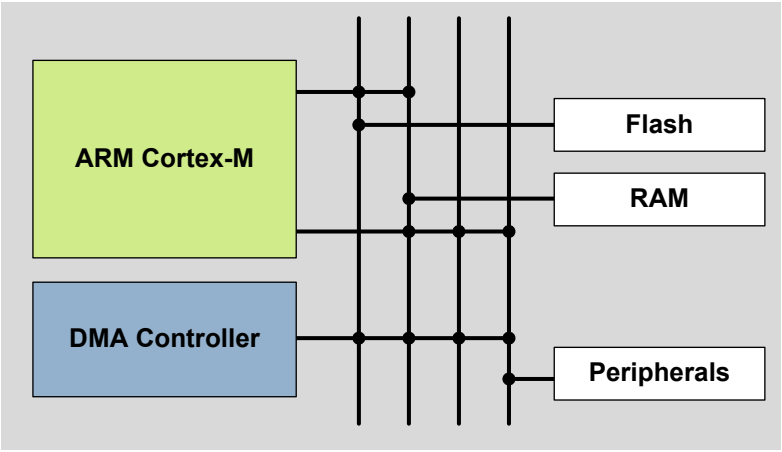
See the individual peripheral chapters for more information on interrupt function.

IRQ #	Name	Source(s)
0*	CRYPTOACC	CRYPTOACC.CRYPTOMASTER
1*	TRNG	CRYPTOACC.TRNG
2*	PKE	CRYPTOACC.PKE
3*	SMU_SECURE	SMU.SECURE
4*	SMU_PRIVILEGED	SMU.PRIVILEGED
5*	SMU_NS_PRIVILEGED	
6*	EMU	EMU.MAIN
7*	TIMER0	TIMER0.MAIN
8*	TIMER1	TIMER1.MAIN
9*	TIMER2	TIMER2.MAIN
10*	TIMER3	TIMER3.MAIN
11*	TIMER4	TIMER4.MAIN
12*	RTCC	RTCC.MAIN
13*	USART0_RX	USART0.RX
14*	USART0_TX	USART0.TX
15*	USART1_RX	USART1.RX
16*	USART1_TX	USART1.TX
17*	ICACHE0	ICACHE0.MAIN
18*	BURTC	BURTC.MAIN
19*	LETIMER0	LETIMER0.MAIN
20*	SYSCFG	SYSCFG.MAIN
21*	LDMA	LDMA.MAIN
22*	LFXO	LFXO.MAIN
23*	LFRCO	LFRCO.MAIN
24*	ULFRCO	ULFRCO.MAIN
25*	GPIO_ODD	GPIO.ODD
26*	GPIO_EVEN	GPIO.EVEN
27*	I2C0	I2C0.irq
28*	I2C1	I2C1.irq
29*	EMUDG	EMU.DG
30*	EMUSE	EMU.SE
31*	AGC	AGC.MAIN
32*	BUFC	BUFC.MAIN
33*	FRC_PRI	FRC.PRI

IRQ #	Name	Source(s)
34*	FRC	FRC.MAIN
35*	MODEM	MODEM.MAIN
36*	PROTIMER	PROTIMER.MAIN
37*	RAC_RSM	RAC.RSM
38*	RAC_SEQ	RAC.SEQ
39*	RDMAILBOX	
40*	RFSENSE	
41*	PRORTC	PRORTC.MAIN
42*	SYNTH	SYNTH.MAIN
43*	WDOG0	WDOG0.MAIN
44*	HFXO0	HFXO0.MAIN
45*	HFRCO0	HFRCO0.MAIN
46*	CMU	CMU.MAIN
47*	AES	RADIOAES.MAIN
48*	IADC	IADC0.MAIN
49*	MSC	MSC.irq_imem
50*	DPLL0	DPLL0.MAIN
51*	PDM	
52*	SW0	SYSCFG.SW0
53*	SW1	SYSCFG.SW1
54*	SW2	SYSCFG.SW2
55*	SW3	SYSCFG.SW3
56*	KERNEL0	
57*	KERNEL1	
58*	M33CTI0	CORE.CTI0
59*	M33CTI1	CORE.CTI1
60*	EMUEFP	
61*	DCDC	
62*	EUART0_RX	
63*	EUART0_TX	

4. Memory and Bus System





Quick Facts

What?

A low latency memory system including low energy Flash and RAM with data retention which makes the low energy modes attractive.

Why?

RAM retention reduces the need for storing data in Flash and enables frequent use of the ultra low energy modes EM2 and EM3.

How?

Low energy and non-volatile Flash memory stores program and application data in all energy modes and can easily be reprogrammed in system. Low leakage RAM with data retention in EM0 to EM3 removes the data restore time penalty, and the DMA ensures fast autonomous transfers with predictable response time.

4.1 Introduction

The EFR32xG22 contains a set of AMBA buses which move data between peripherals, memory, and the CPU. All memories and register interfaces are memory mapped into a unified address space.

4.2 Functional Description

The internal memory segments of the Cortex-M33 are mapped into the system memory map as shown by Figure 4.1 System Address Space with Core and Code Space Listing on page 37.

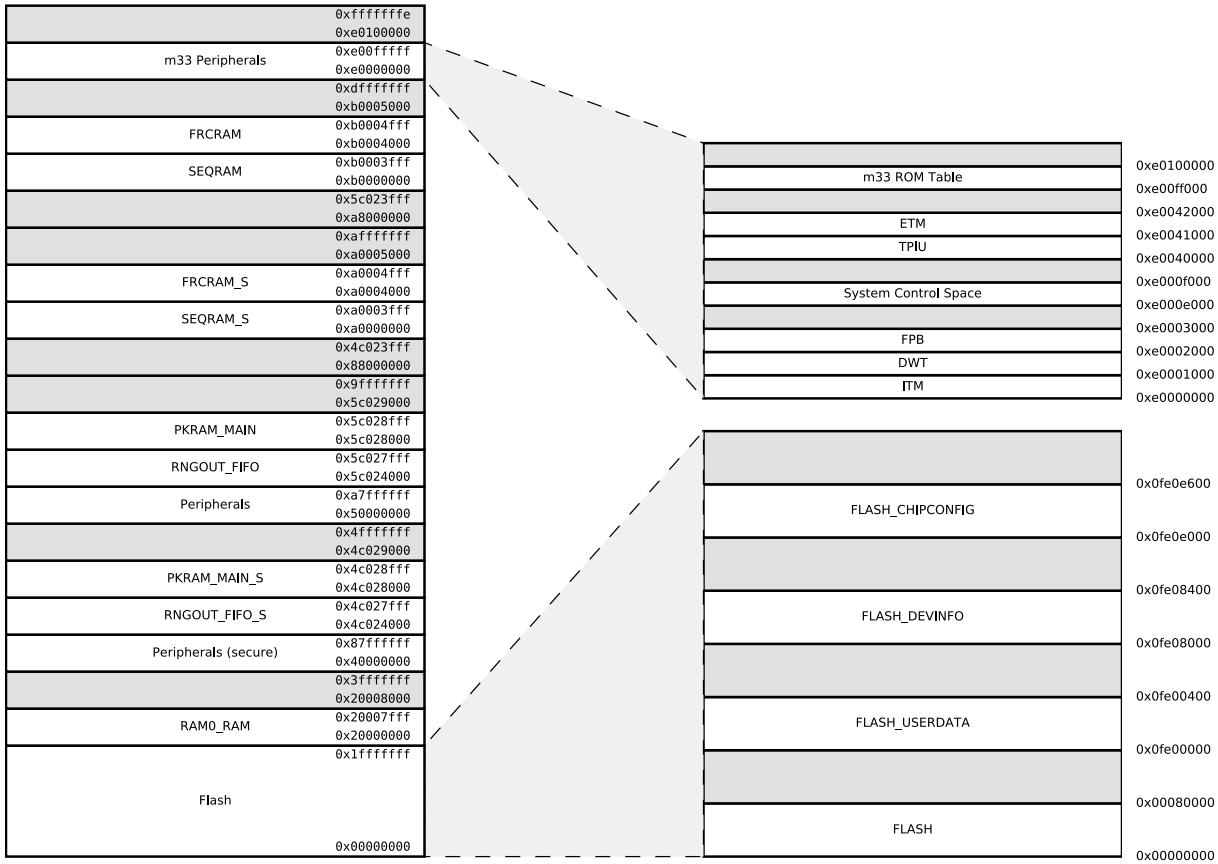


Figure 4.1. System Address Space with Core and Code Space Listing

Flash for the main program memory (CODE) is located at address 0x00000000 in the memory map of the EFR32xG22.

SRAM for the main data memory (RAM) is located at address 0x20000000 in the memory map of the EFR32xG22. When running code located in RAM, the Cortex-M33 uses the System bus interface to fetch instructions. This results in reduced performance as the Cortex-M33 accesses stack, other data in SRAM and peripherals using the System bus interface.

The Sequencer RAM (SEQRAM) is located at address 0xA0000000 and is used by the Sequencer for both instructions and data. This RAM is also available for general use if not required by the RF subsystem.

4.2.1 Bus Matrix

A multilayer AMBA AHB bus matrix connects the master bus interfaces to the AHB slaves. The bus matrix allows several AHB slaves to be accessed simultaneously. An AMBA APB interface is used for the peripherals, which are accessed through an AHB-to-APB bridge connected to the AHB bus matrix.

The CPU has two AHB bus masters (Code and System) so that it may retrieve instructions and data in parallel. The Code master is used to access all memory below address 0x20000000 and the System master access addresses 0x20000000 and above.

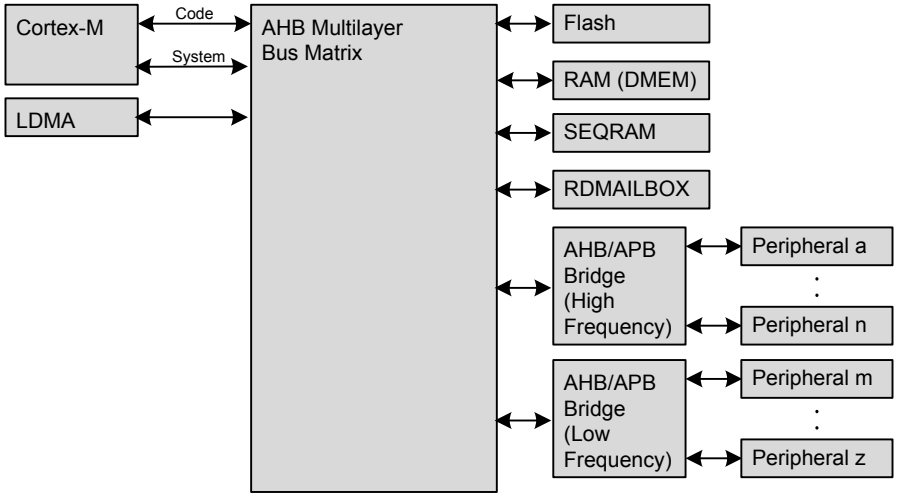


Figure 4.2. EFR32xG22 Bus System

4.2.1.1 Arbitration

The Bus Matrix uses a round-robin arbitration algorithm which enables high throughput and low latency, while starvation of simultaneous accesses to the same bus slave are eliminated. Round-robin does not assign a fixed priority to each bus master. The arbiter does not insert any bus wait-states during peak interaction. However, one wait state is inserted for master accesses occurring after a prolonged inactive time. This wait state allows for increased power efficiency during master idle time.

4.2.1.2 Bus Faults

System accesses from the core can receive a bus fault in the following condition(s):

- The core attempts to access an address that is not assigned to any peripheral or other system device. These faults can be enabled or disabled by setting the ADDRFAULTEN bit in the SYSCFG_CTRL register.
- The core attempts to access a peripheral register that is LOCKED.
- The core attempts to access a peripheral or system device that has its clock disabled. This fault can be enabled or disabled by setting the ADDRFAULTEN bit in the SYSCFG_CTRL register.
- System RAM controller or RADIO RAM controller detects a 2bit ECC error. These faults can be enabled or disabled by setting the RAMECCERRFAULTEN bit in the SYSCFG_CTRL register
- Registers with synchronization requirements may generate bus faults if accessed incorrectly. See 4.2.4.4 Peripheral Access Performance for more details on register access types. In particular the following actions can cause bus faults:
  - Config register written while peripheral enabled.
  - Sync register written while peripheral disabled
  - LfSync register written while a previous write is pending
  - Peripheral disabled while any LfSync write is pending

In addition to any condition-specific bus fault control bits, the bus fault interrupt itself can be enabled or disabled in the same way as all other internal core interrupts.

4.2.2 Flash

The Flash retains data in any state and typically stores the application code and special user data. The Flash memory is typically programmed through the debug interface, but can also be erased and written to from software.

- Up to 512 kB of memory
- Page size of 8 KB (minimum erase unit)
- Minimum 10k erase cycles endurance
- Greater than 10 years data retention at 85°C
- Lock registers for memory protection
- Data retention in any state

4.2.3 SRAM

The primary task of the SRAM memory is to store application data. Additionally, it is possible to execute instructions from SRAM, and the DMA may be set up to transfer data between the SRAM, flash and peripherals.

The device contains several blocks of SRAM for various purposes including general data memory (DRAM) and various RF subsystem rams (SEQRAM, FRCRAM). For more detailed information see 6. MSC - Memory System Controller .

- Up to 32 kB of memory (RAM)
- RAM blocks may be powered down when not in use
- Data retention of the entire memory or selected banks in EM2 and EM3

Note: Root code requires some RAM storage during a system reset. The RAM used by root code is located at the top of the DRAM memory space. If the user application requires RAM that persists through reset, it is recommended to use a statically allocated section at the bottom of the SRAM memory space (address 0x20000000).

4.2.4 Peripherals

The peripherals are mapped into the peripheral memory segment, each with a fixed size address range shown in the 4.2.4.1 Peripheral Map

4.2.4.1 Peripheral Map

This table shows the address range for each peripheral. In addition it shows the lowest energy mode in which the peripheral is powered. Note that EM3 is defined as EM2 with all clocks disabled. Therefore all peripherals powered in EM2 are also powered in EM3 but may not function if they require a running clock.

See the individual peripheral chapters for more information on low power operation.

Address Range	Module Name	Power Domain
0x40004000 - 0x40007FFF	EMU	EM2.A
0x40008000 - 0x4000BFFF	CMU	EM2.B
0x4000C000 - 0x4000FFFF	HFXO0	EM1
0x40010000 - 0x40013FFF	HFRCO0	EM1
0x40018000 - 0x4001BFFF	FSRCO	EM4
0x4001C000 - 0x4001FFFF	DPLL0	EM1
0x40020000 - 0x40023FFF	LFXO	EM4
0x40024000 - 0x40027FFF	LFRCO	EM2.C / EM4[1]
0x40028000 - 0x4002BFFF	ULFRCO	EM4
0x40030000 - 0x40033FFF	MSC	EM1
0x40034000 - 0x40037FFF	ICACHE0	EM1
0x40038000 - 0x4003BFFF	PRS	EM2.B
0x4003C000 - 0x4003FFFF	GPIO	EM2.B
0x40040000 - 0x40043FFF	LDMA	EM1
0x40044000 - 0x40047FFF	LDMAXBAR	EM1
0x40048000 - 0x4004BFFF	TIMER0	EM1
0x4004C000 - 0x4004FFFF	TIMER1	EM1
0x40050000 - 0x40053FFF	TIMER2	EM1
0x40054000 - 0x40057FFF	TIMER3	EM1
0x40058000 - 0x4005BFFF	TIMER4	EM1
0x4005C000 - 0x4005FFFF	USART0	EM1
0x40060000 - 0x40063FFF	USART1	EM1
0x40064000 - 0x40067FFF	BURTC	EM4
0x40068000 - 0x4006BFFF	I2C1	EM1
0x40078000 - 0x4007BFFF	SYSCFG	EM1
0x4007C000 - 0x4007FFFF	SYSCFG	EM1
0x40080000 - 0x40083FFF	BURAM	EM4
0x40088000 - 0x4008BFFF	GPCRC	EM1
0x40094000 - 0x40097FFF	DCDC	EM2.B
0x40098000 - 0x4009BFFF	PDM	EM1
0x4009C000 - 0x4009FFFF	RFSENSE	EM4
0x44000000 - 0x440007FF	RADIOAES	EM1
0x44008000 - 0x4400BFFF	SMU	EM1

Address Range	Module Name	Power Domain
0x4400C000 - 0x4400FFFF	SMU	EM1
0x48000000 - 0x48003FFF	RTCC	EM2.A
0x4A000000 - 0x4A003FFF	LETIMER0	EM2.B
0x4A004000 - 0x4A007FFF	IADC0	EM2.B
0x4A010000 - 0x4A013FFF	I2C0	EM2.B
0x4A018000 - 0x4A01BFFF	WDOG0	EM2.B
0x4A020000 - 0x4A023FFF	AMUXCP0	EM2.B
0x4A030000 - 0x4A033FFF	EUART0	EM2.B
0x4C020000 - 0x4C0207FF	CRYPTOACC	EM1
0x4C021000 - 0x4C02107F	CRYPTOACC	EM1
0x4C022000 - 0x4C02201F	CRYPTOACC	EM1
0x50004000 - 0x50007FFF	EMU_NS	EM2.A
0x50008000 - 0x5000BFFF	CMU_NS	EM2.B
0x5000C000 - 0x5000FFFF	HFXO0_NS	EM1
0x50010000 - 0x50013FFF	HFRCO0_NS	EM1
0x50018000 - 0x5001BFFF	FSRCO_NS	EM4
0x5001C000 - 0x5001FFFF	DPLL0_NS	EM1
0x50020000 - 0x50023FFF	LFXO_NS	EM4
0x50024000 - 0x50027FFF	LFRCO_NS	EM2.C / EM4[1]
0x50028000 - 0x5002BFFF	ULFRCO_NS	EM4
0x50030000 - 0x50033FFF	MSC_NS	EM1
0x50034000 - 0x50037FFF	ICACHE0_NS	EM1
0x50038000 - 0x5003BFFF	PRS_NS	EM2.B
0x5003C000 - 0x5003FFFF	GPIO_NS	EM2.B
0x50040000 - 0x50043FFF	LDMA_NS	EM1
0x50044000 - 0x50047FFF	LDMAXBAR_NS	EM1
0x50048000 - 0x5004BFFF	TIMER0_NS	EM1
0x5004C000 - 0x5004FFFF	TIMER1_NS	EM1
0x50050000 - 0x50053FFF	TIMER2_NS	EM1
0x50054000 - 0x50057FFF	TIMER3_NS	EM1
0x50058000 - 0x5005BFFF	TIMER4_NS	EM1
0x5005C000 - 0x5005FFFF	USART0_NS	EM1
0x50060000 - 0x50063FFF	USART1_NS	EM1
0x50064000 - 0x50067FFF	BURTC_NS	EM4
0x50068000 - 0x5006BFFF	I2C1_NS	EM1
0x50078000 - 0x5007BFFF	SYSCFG_NS	EM1
0x5007C000 - 0x5007FFFF	SYSCFG_NS	EM1

Address Range	Module Name	Power Domain
0x50080000 - 0x50083FFF	BURAM_NS	EM4
0x50088000 - 0x5008BFFF	GPCRC_NS	EM1
0x50094000 - 0x50097FFF	DCDC_NS	EM2.B
0x50098000 - 0x5009BFFF	PDM_NS	EM1
0x5009C000 - 0x5009FFFF	RFSENSE_NS	EM4
0x54000000 - 0x540007FF	RADIOAES_NS	EM1
0x54008000 - 0x5400BFFF	SMU_NS	EM1
0x5400C000 - 0x5400FFFF	SMU_NS	EM1
0x58000000 - 0x58003FFF	RTCC_NS	EM2.A
0x5A000000 - 0x5A003FFF	LETIMER0_NS	EM2.B
0x5A004000 - 0x5A007FFF	IADC0_NS	EM2.B
0x5A010000 - 0x5A013FFF	I2C0_NS	EM2.B
0x5A018000 - 0x5A01BFFF	WDOG0_NS	EM2.B
0x5A020000 - 0x5A023FFF	AMUXCP0_NS	EM2.B
0x5A030000 - 0x5A033FFF	EUART0_NS	EM2.B
0x5C020000 - 0x5C0207FF	CRYPTOACC_NS	EM1
0x5C021000 - 0x5C02107F	CRYPTOACC_NS	EM1
0x5C022000 - 0x5C02201F	CRYPTOACC_NS	EM1
0xA8004000 - 0xA8007FFF	FRC	EM1
0xA800C000 - 0xA800FFFF	AGC	EM1
0xA8010000 - 0xA8013FFF	RFCRC	EM1
0xA8014000 - 0xA8017FFF	MODEM	EM1
0xA8018000 - 0xA801BFFF	SYNTH	EM1
0xA801C000 - 0xA801FFFF	PROTIMER	EM1
0xA8020000 - 0xA8023FFF	RAC	EM1
0xA8028000 - 0xA802BFFF	RDMAILBOX0	EM1
0xA802C000 - 0xA802FFFF	RDMAILBOX1	EM1
0xAA000000 - 0xAA003FFF	BUFC	EM1
0xB8004000 - 0xB8007FFF	FRC_NS	EM1
0xB800C000 - 0xB800FFFF	AGC_NS	EM1
0xB8010000 - 0xB8013FFF	RFCRC_NS	EM1
0xB8014000 - 0xB8017FFF	MODEM_NS	EM1
0xB8018000 - 0xB801BFFF	SYNTH_NS	EM1
0xB801C000 - 0xB801FFFF	PROTIMER_NS	EM1
0xB8020000 - 0xB8023FFF	RAC_NS	EM1
0xB8028000 - 0xB802BFFF	RDMAILBOX0_NS	EM1
0xB802C000 - 0xB802FFFF	RDMAILBOX1_NS	EM1

Address Range	Module Name	Power Domain
0xBA000000 - 0xBA003FFF	BUFC_NS	EM1

Note:
1. LFRCO requires EM0, EM1, or EM2 when operating in precision mode, but can operate down to EM4 when precision mode is not used.

4.2.4.2 Peripheral non-word access behavior

When writing to peripheral registers, all accesses are treated as 32-bit accesses. This means that writes to a register need to be large enough to cover all bits of register, otherwise, any uncovered bits may become corrupted from the partial-word transfer. Thus, the safest practice is to always do 32-bit writes to peripheral registers.

When reading, there is generally no issue with partial word accesses, however, note that any read action (e.g. FIFO popping) will be triggered regardless of whether the actual FIFO bit-field was included in the transfer size.

4.2.4.3 Peripheral Bit Set and Clear

The EFR32xG22 supports bit set, bit clear, and bit toggle access to most peripheral registers. The bit set and bit clear functionality (also called Bit Access) enables modification of bit fields without the need to perform a read-modify-write. Also, the operation is contained within a single bus access. Bit access registers and their addresses are shown in the register map for each peripheral. Peripherals with no _SET, _CLR, or _TGL registers in the register map to not support these functions.

Each register with Bit Set functionality will have a _SET register. Whenever a bit in the SET register is written to a 1 the corresponding bit in its target register is set. The SET register is located at TARGET + 0x1000 where TARGET is the address of the target register and has the same name as the target register with '_SET' appended.

Each register with Bit Clear functionality will have a CLR register. Whenever a bit in the CLR register is written to a 1 the corresponding bit in its target register is cleared. The CLR register is located at TARGET + 0x2000 where TARGET is the address of the target register and has the same name as the target register with '_CLR' appended.

Each register with Bit Toggle functionality will have a TGL register. Whenever a bit in the TGL register is written to a 1 the corresponding bit in its target register is inverted. The TGL register is located at TARGET + 0x3000 where TARGET is the address of the target register and has the same name as the target register with '_TGL' appended.

Note: It is possible to combine bit clear and bit set operations in order to arbitrarily modify multi-bit register fields without affecting other fields in the same register. In this case, care should be taken to ensure that the field does not have intermediate values that can lead to erroneous behavior. For example, if bit clear and bit set operations are used to change an analog tuning register field from 0x2 to 0x4 by clearing bit 1 and then setting bit 2, the field would take on a value of zero for short time. If the analog module is active at the time, this could lead to undesired behavior.

4.2.4.4 Peripheral Access Performance

The Cortex-M33, DMA Controller, and peripherals run on clocks which can be pre-scaled separately. Clocks and pre-scaling are described in more detail in 8. CMU - Clock Management Unit. This section describes the access performance for a peripheral register based on its frequency relative to the CPUCLK frequency and its access type. For this discussion, PERCLK refers to a selected peripheral's clock frequency and CPUCLK refers to the core's clock frequency.

The type of each register in a peripheral is indicated in the 'Access' column of the peripherals register table. Register types are: ENABLE, CONFIG, SYNC, LFSYNC, and INTFLAG. If not type is listed then the register is a Generic register.

4.2.4.4.1 Generic Registers

Registers with no type listed are generic registers. They may be read or written to at any time. Access will not stall the CPU.

4.2.4.4.2 CONFIG Registers

CONFIG Registers contain configuration that does not change during peripheral operation.

CONFIG registers may only be written when a peripheral is disabled. Writing to a CONFIG register when a peripheral is enabled will result in a BUSFAULT. CONFIG register writes will not stall the CPU.

CONFIG registers may be read at any time. Reads will not stall the CPU.

4.2.4.4.3 SYNC Registers

SYNC registers are used to communicate with running high-speed peripherals where PERCLK is expected to be either higher or marginally slower (within an order of magnitude) than CPUCLK. For example a timer running at 76.8 MHz when the core is at 38.4 MHz or at 9.6 MHz when the core is 76.8 MHz. In this case CPU stalls of several PERCLK cycles do not significantly impact overall system performance in most systems.

SYNC registers may only be written to when the peripheral is enabled. Writing to a SYNC register when a peripheral is disabled will result in a BUSFAULT. A write will take several (2 - 3) PERCLK cycles to complete (take effect) during which time the entire module will be in a pending state. If a SYNC register is written to while the peripheral is already in a pending state, the CPU is stalled until the previous write finishes. If a SYNC register is written to while the peripheral is not in a pending state, the CPU is not stalled.

SYNC registers may be read at any time. If a SYNC register is read while the peripheral is disabled, the CPU is not stalled. If a SYNC register is read while the peripheral is enabled, the CPU will be stalled for several (2 -3) PERCLK cycles while up to date values are retrieved from the peripheral.

4.2.4.4.4 LFSYNC Registers

LFSYNC registers are used to communicate with running low frequency peripherals where PERCLK is expected to be much lower than the CPU clock and synchronization delays may be long. For example, a LETIMER running at 32 kHz when the core is at 76.8 MHz. In this case CPU stalls of several PERCLK cycles represent a significant blockage of the CPU and need to be avoided whenever possible. LFSYNC registers accommodate this by allowing the CPU to write the register and continue to do other work while the value is synchronized.

LFSYNC registers may be written at any time. A write will take several (3 -4) PERCLK cycles to complete during which the register will be in a pending state. If a LFSYNC register is written to while it is in a pending state, a BUSFAULT will occur. Each LFSYNC register has a status bit indicating if it is currently pending.

LFSYNC registers may be read at any time. The CPU is never stalled on a read. If a LFSYNC register is read while pending, the pending (recently written) data will be returned even though it has not yet taken effect. Software may use the busy status bit to determine if the read value has been applied to the hardware.

4.2.4.4.5 ENABLE Registers

ENABLE registers contain the enable bit for a peripheral.

ENABLE registers may be written at any time. When the peripheral is enabled it takes some time for the enable to take effect during which time the module is pending. Peripherals will be in the pending state for a few (2 - 3) PERCLK cycles when first enabled. Since the clock source for the peripheral may not be running before the peripheral is enabled, the start up time for the clock source may increase the pending time. See EFR32xG22 Wireless Gecko for more information on on-demand clock sources.

Disabling a high frequency module will stall the CPU until all pending SYNC writes have completed and any pending enable has completed. If the module is fully enabled and no SYNC writes are pending, the disable will be instantaneous. Disabling low frequency peripheral which a LFSYNC is pending will result in a bus fault. Disabling a low frequency peripherals while an enable is still pending causes no CPU stall.

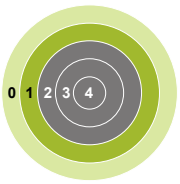
ENABLE registers may be read at any time.

4.2.4.4.6 INTFLAG Registers

INTFLAG registers contain interrupt flags. To set or clear an interrupt flag, the _SET or _CLR register alias must be used. Writing directly to the INTFLAG register will have no effect.

Note that for an interrupt to occur when a flag is set the IRQ must be enabled in the NVIC.

5. Radio Transceiver



Quick Facts

What?

The Radio Transceiver provides access to transmit and receive data, radio settings and control interface.

Why?

The Radio Transceiver enables the user to communicate using a wide range of data rates, modulation and frame formats.

How?

Dynamic or fixed frame lengths, optional address recognition, flexible CRC and crypto schemes makes the EFR32 Series 2 perfectly suit any application using low or medium data rate radio communication.

5.1 Introduction

The Radio Transceiver of the EFR32 Series 2 enables the user to control a wide range of settings and options for tailoring radio operation precisely to the users need. It provides access to the transmit and receive data buffers and supports both dynamic and static frame lengths, as well as automatic address filtering and CRC insertion/verification.

As seen in the Radio Overview illustration (Figure 5.1 Radio Overview on page 46), the radio consists of several modules all responsible for specific tasks. Please refer to the abbreviations section (Appendix 1. Abbreviations) for a comprehensive description of acronyms.

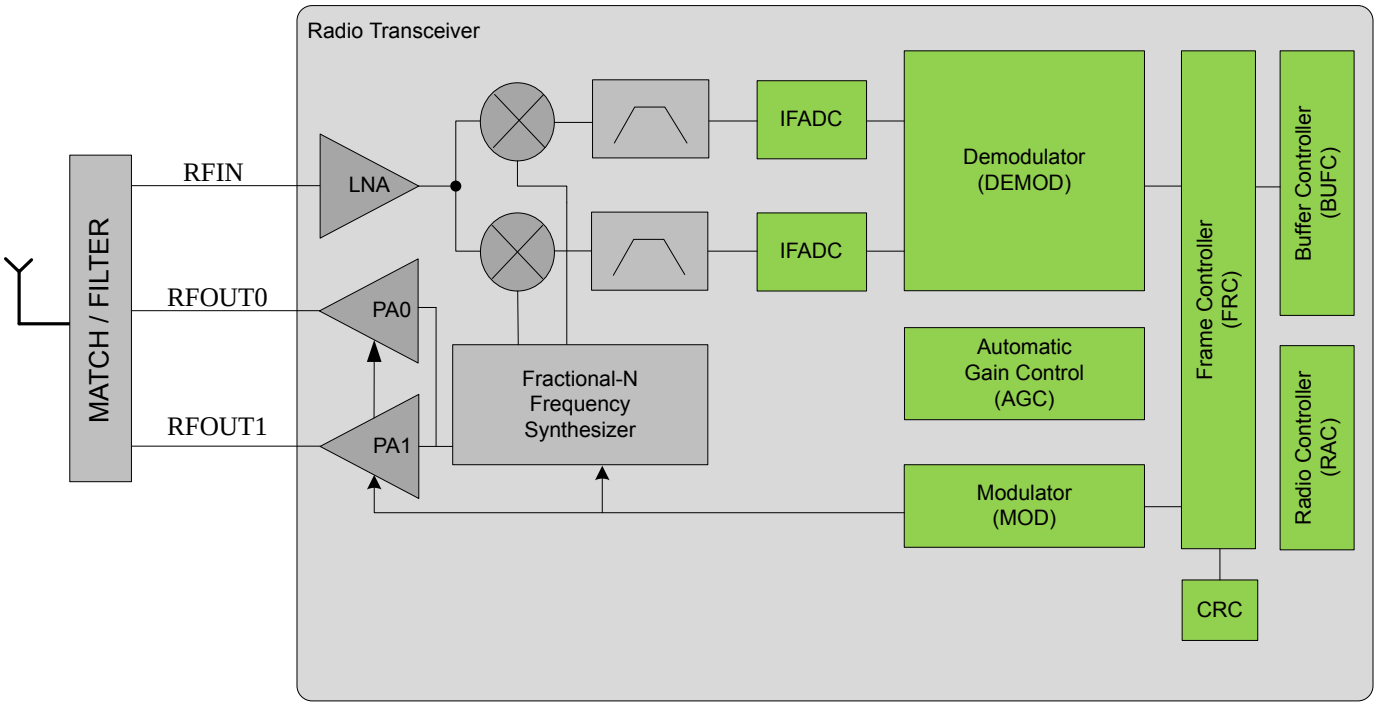


Figure 5.1. Radio Overview

During transmission (TX), the Radio Controller enables the SYNTH, Modulator and PA. The Modulator requests data from the Frame Controller, which reads data from a buffer. Based upon modulation format and data to send, the Modulator manipulates the SYNTH to output the correct frequency and phase. When the whole frame has been transmitted, the radio can automatically switch to receive mode.

In receive mode (RX), the radio controller enables the LNA, SYNTH, Mixer, ADC and Demodulator. The Demodulator searches for valid frames according to modulation format and data rate. If a frame is detected, the demodulated data is handed to the Frame Controller, which stores the data in the Buffer. When the complete frame has been received (determined by the Frame Controller), it is possible to either go to TX or stay in RX to search for a new frame.

The Radio Transceiver interface is accessible through software drivers provided by Silicon Labs.

5.1.1 RF Frequency Synthesizer

The Fractional-N RF Frequency Synthesizer (SYNTH) provides a low phase noise LO signal to be used in both receive and transmit modes.

The capabilities of the SYNTH include:

- High performance, low phase noise
- Fast frequency settling
- Fast and fully automated calibration
- Sub 100 Hz frequency resolution across the supported frequency bands

5.1.2 Modulation Modes

EFR32xG22 supports a wide range of modulation modes in transmit and receive:

- 2-FSK, 2-GFSK, 4-FSK, MSK, GMSK, O-QPSK with half-sine shaping, ASK / OOK, DBPSK TX
- NRZ or Manchester support
- UART mode over air for legacy protocols
- Baudrates ranging from below 100 Baud/s to 2 MBaud/s, allowing data rates up to 4 MBit/s
- Configurable frequency deviation
- Configurable Direct Sequence Spread Spectrum (DSSS), with spread sequences up to 32 chips encoding up to 4 information bits
- Configurable 4-FSK symbol encoding

5.1.3 Transmit Mode

In transmit mode EFR32xG22 performs the following functionality:

- Automatic PA power ramping during the start and end of a frame transmit
- Programmable output power
- Optional preamble and synchronization word insertion
- Accurate transmit frame timing to support time synchronized radio protocols
- Optional Carrier Sense Multiple Access - Collision Avoidance (CSMA-CA) or Listen Before Talk (LBT) hardware support
- Integrated transmit test modes, as described in 5.1.13 RF Test Modes

5.1.4 Receive Mode

In receive mode EFR32xG22 performs the following functionality:

- A single-ended (2.4 GHz) LNA amplifies the input RF signal. The amplified signal is then mixed to a low-IF signal through the quadrature down-coversion mixer. Further signal filtering is performed before conversion to a digital signal through the I/Q ADC.
- Digitally configurable receiver bandwidth from 100 Hz to 2.5 MHz
- Timing recovery on received data, including simultaneous support for two different frame synchronization words
- Automatic frequency offset compensation, to compensate for carrier frequency offset between the transmitter and receiver
- Support for a wide range of modulation formats as described in section 5.1.2 Modulation Modes

5.1.5 Data Buffering

EFR32xG22 supports buffered transmit and receive modes through its buffer controller (BUFC), with four individually configurable buffers. The BUFC uses the system RAM as storage, and each buffer can be individually configured with parameters such as:

- Buffer size
- Buffer interrupt thresholds
- Buffer RAM location
- Overflow and underflow detection

In receive mode, data following frame synchronization is moved directly from the demodulator to the buffer storage.

In transmit mode, data following the inserted preamble and synchronization word is moved directly from the buffer storage to the modulator.

5.1.6 Unbuffered Data Transfer

For most system designs it is recommended to use the data buffering within EFR32xG22 to provide a convenient user interface.

In unbuffered data transfer modes the hardware support provided by EFR32xG22 to perform preamble and frame synchronization insertion in transmit mode and detection in receive mode can still optionally be used.

5.1.7 Frame Format Support

EFR32xG22 has an extensive support for frame handling in transmit and receive modes, which allows effective handling of even advanced protocols. The frame format support is controlled by the Frame Controller (FRC). The support includes:

- Preamble and frame synchronization inserted into transmitted frames
- Full frame synchronization of received frames
- Simple address matching of received frames in hardware, further configurable address and frame filtering supported through sequencer
- Support for variable length frames
- Automated CRC calculation and verification
- Configurable bit ordering, with the most or least significant bit transmitted and received first

5.1.8 Hardware CRC Support

EFR32xG22 supports a configurable CRC generation in transmit and verification in receive mode:

- 8, 16, 24 or 32 bit CRC value
- Configurable polynomial and initialization value
- Optional inversion of CRC value over air
- Configurable CRC byte ordering
- Support for multiple CRC values calculated and verified per transmitted or received frame
- The CRC module is typically controlled by the Frame Controller (FRC) for in-line operations in transmit and receive modes. Alternatively, the CRC module may be accessed directly from software to calculate and verify CRC data.

5.1.9 Convolutional Encoding / Decoding

EFR32xG22 includes hardware support for convolutional encoding and decoding, for forward error correction (FEC). This feature is performed by the Frame Controller (FRC) module:

- Constraint length configurable up to 7, for the highest robustness
- Configurable puncturing, to achieve rates between 1/2 rate and full rate
- Configurable soft decision or hard decision decoding
- Convolutional coding may be used together with the symbol interleaver to improve robustness against burst errors

5.1.10 Binary Block Encoding / Decoding

EFR32xG22 includes hardware support for binary block encoding and decoding, both performed real-time in the the transmit and receive path. This is performed in the Frame Controller (FRC) module:

The block coding works on blocks of up to 16 bits of data and adds parity bits to be capable of single or multiple bit corrections by the receiver.

- One or more parity bits can be added and verified
- Bit error correction
- Lookup-codes can be used to implement virtually any block coding scheme

5.1.11 Data Encryption and Authentication

EFR32xG22 has hardware support for AES encryption, decryption and authentication modes. These security operations can be performed on data in RAM or any data buffer, without further CPU intervention. The key size is 128 bits.

AES modes of operations directly supported by the EFR32xG22 hardware are listed in Table 5.1 AES modes of operation with hardware support on page 49. In addition to these modes, other modes can also be implemented by using combinations of modes. For example, the CCM mode can be implemented using the CTR and CBC-MAC modes in combination.

Table 5.1. AES modes of operation with hardware support

AES Mode	Encryption / Decryption	Authentication	Comment
ECB	Yes	-	Electronic Code Book
CTR	Yes	-	Counter mode
CCM	Yes	Yes	Counter with CBC-MAC
CCM*	Yes	Yes	CCM with encryption-only and integrity-only capabilities
GCM	Yes	Yes	Galois Counter Mode
CBC	Yes	-	Cipher Block Chaining
CBC-MAC	-	Yes	Cipher Block Chaining, Message Authentication Code
CMAC	-	Yes	Cipher-based MAC
CFB	Yes	-	Cipher Feedback
OFB	Yes	-	Output Feedback

The Cryptographic Acceleration module can operate directly on data buffers provided by the buffer controller (BUFC) module. It is also possible to provide data directly from the embedded Cortex-M33 or via DMA.

5.1.12 RFSENSE

The RFSENSE block on the EFR32xG22 is an ultra-low energy RF signal detector which provides wake-on-RF capabilites from any energy mode. The system can remain in low energy modes such as EM2 or EM4 for long durations while continuously monitoring for a valid wake condition. RFSENSE can operate as a selective On Off Keying (OOK) detector, or a simple RF energy detector.

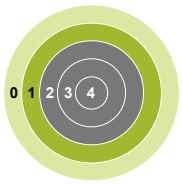
- Selective OOK pattern detection:
  - 1 kHz OOK symbol rate, manchester encoded (0.5 kHz bit rate)
  - Configurable 4/8-bit preamble length with fixed 1010 pattern
  - Configurable 8/16/24/32-bit sync word length with fully programmable pattern
- Simple RF energy threshold detection:
  - Programmable RF energy trip point
  - Configurable energy averaging duration

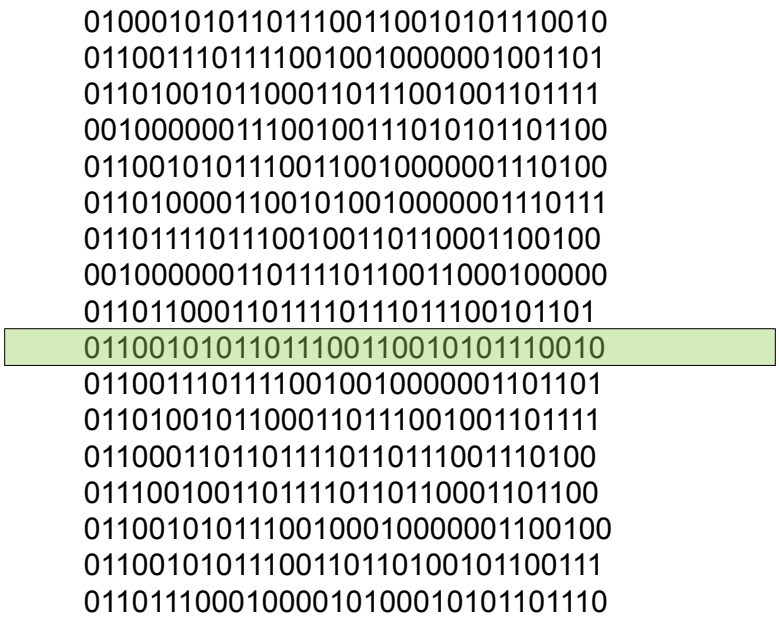
5.1.13 RF Test Modes

EFR32xG22 supports a wide range of RF test modes typically used for characterization and regulation compliance testing, including:

- Unmodulated carrier transmit
- Modulated carrier transmit, with internal configurable pseudo random data generator
- Continuous data reception for Bit Error Rate (BER) measurements
- Storing of raw receiver data to RAM
- Transmit of raw frequency data from RAM

6. MSC - Memory System Controller





Quick Facts

What?

The user can perform flash memory read, read configuration, and write operations through the Memory System Controller (MSC). SRAM operation may be configured though System Configuration (SYSCFG).

Why?

The MSC allows the application code and user data to be stored in non-volatile flash memory. Certain memory system functions, such as program memory wait-states and flash lock bits are configured from the MSC peripheral register interface, giving the developer the ability to dynamically customize the memory system performance, security level, energy consumption and error handling capabilities to the requirements at hand.

How?

The MSC integrates a low-energy flash IP with a charge pump, enabling minimum energy consumption while eliminating the need for external programming voltage to erase the memory. An easy to use write and erase interface is supported by an internal, fixed-frequency oscillator and autonomous flash timing and control reduces software complexity while not using other timer resources.

A highly efficient low energy instruction cache reduces the number of flash reads significantly, thus saving energy. Performance is also improved when wait-states are used, since many of the wait-states are eliminated. Built-in performance counters can be used to measure the efficiency of the instruction cache.

Instruction prefetcher improves program execution performance by reducing the number of wait-state cycles needed.

6.1 Introduction

The Memory System Controller (MSC) is the program memory unit of the EFR32xG22 microcontroller. The flash memory is readable and writable from both the Cortex-M33 and DMA. The flash memory is divided into two blocks: the main block and the information block. Program code is normally written to the main block. The information block is available for special user data. There is also a read-only page in the information block containing system and device calibration data. Flash read and write operations are supported in energy modes EM0 and EM1.

6.2 Features

- AHB read interface
  - Scalable access performance to optimize the Cortex-M33 code interface
    - Advanced energy optimization functionality
      - Conditional branch target prefetch suppression
      - Cortex-M33 disfolding of if-then (IT) blocks
      - Instruction Cache
      - Instruction Prefetch
  - DMA read support in EM0 and EM1
- Command and status interface
  - Flash write and erase
    - Accessible from Cortex-M33 in EM0
    - DMA write support in EM0 and EM1
  - Core clock independent Flash timing
    - Internal oscillator and internal timers for precise and autonomous Flash timing
      - General purpose timers are not occupied during Flash erase and write operations
      - No special time scaling registers needed
  - Configurable interrupt erase abort
    - Improved interrupt predictability
  - Memory and bus fault control
- Security features
  - Lockable debug access
  - Page lock registers
  - SW Mass erase and User Data lock bits
- End-of-write and end-of-erase interrupts

6.3 Functional Description

The size of the main flash block is device dependent. The largest size available is 512 kB (64 pages). The information block has 1 KB available for user data. The information block also contains chip configuration data located in a reserved area. The main block is mapped to address 0x00000000 and the information block is mapped to address 0x0FE00000. Table 6.1 MSC Flash Memory Mapping on page 51 outlines how the flash is mapped in the memory space. All flash memory is organized into 8 KB pages.

Table 6.1. MSC Flash Memory Mapping

Block	Page	Base address	Write/Erase by...	Software Readable?	Purpose/Name	Size
Main	0	0x00000000	Software, debug	Yes	User code and data	16 KB - 512 kB
	1	0x00002000	Software, debug	Yes		
	...		Software, debug	Yes		
	63[1]	0x0007E000	Software, debug	Yes		
Information	N/A	0x0FE00000	Software	Yes	User Data (UD)	1 KB
	N/A	0x0FE08000	-	Yes	Device Information (DI)	1 KB

Note:

1. 64 pages for largest device.

6.3.1 Ram Configuration

The SYSCFG module contains controls for configuring the various RAM blocks on the device. Options include enabling EM2/EM3 data retention, ECC, prefetch, and cache. For a complete description see 6.6 SYSCFG - System Configuration.

6.3.2 Instruction Cache

The instruction cache improves the speed and power consumption of the Cortex-M33 by providing fast, low-power access to recently executed instructions. For detailed information see 6.5 ICACHE - Instruction Cache

6.3.3 Device Information (DI) Page

This read-only page holds calibration data from the production test, several unique device IDs, and other part specific information. For a complete description see 6.4 DEVINFO - Device Info Page.

6.3.4 User Data (UD) Page Description

This is the user data page in the information block. The page can be erased and written by software.

This page is written in the same way as any page in the Main user code area.

6.3.5 Bootloader

The EFR32xG22 supports use of the Gecko Bootloader detailed in *UG266: Silicon Labs Gecko Bootloader User's Guide* (https://www.silabs.com/support/resources).To enable bootloader functionality the second stage of the bootloader must be configured and programmed into the first 16KB of flash.

6.3.6 Post-reset Behavior

Calibration values are automatically written to registers by the MSC before application code start-up. The values can also be read from the DI page by software. Other information such as the device ID and production date is also stored in the DI page and is readable from software.

As part of the reset, hardware performs repeated flash reads to determine when flash is fully powered up and available for use by the CPU. PWRUPCKBDFAILCOUNT in MSC_STATUS contains the number of failed reads during the last reset.

6.3.7 Flash Startup

On transitions from EM2/3 to EM0, the flash must be powered up. The time this takes depends on the current operating conditions. To have a deterministic wake time, set STDLY0 in MSC_STARTUP to 0x64 and clear STDLY1, ASTWAIT, STWSEN and STWS. This will result in a 10 us delay before the flash is ready. The system will wake up before this, but the CPU core will stall on the first access to the flash until it is ready. Execute code from RAM or cache to get a faster CPU wake time.

To get a faster flash wake time that depends on the current operating conditions, set STDLY0 to 0x32 and set ASTWAIT in MSC_STARTUP. When configured this way, the system will poll the flash to determine when it is ready, and then start execution.

For the fastest possible wakeup, code may be run with a set of wait-states initially and then automatically switched to normal operation. Set STDLY0 to 0x32, STDLY1 to 0x32, and set ASTWAIT and STWSEN. Then configure STWS in MSC_STARTUP to the number of wait-states to run with. With this setup, execution will begin with the given number of wait-states after 5 uS, and the system will run with reduced throughput due to the wait-states for another 5 us before returning to normal full speed operation

The recommended setting for MSC_STARTUP register is to set STDLY0 to 0x32 for a 5 us wait and set ASTWAIT to one for active sampling. Set STWSEN to zero to bypass second delay period. This provides the best wakeup time without sacrificing power consumption.

Flash wakeup on demand is supported when wakeup from EM2/3 to EM0. Set bit FLASHPWRUPONDEMAND of register EMU_CTRL to enable the power up on demand. When enabled, flash will not be powered up until accessed. In this case it is possible for the MCU to wake, execute out of RAM or cache, and return to sleep mode without ever powering on the Flash. Software can force the flash to power up by writing PWRUP in MSC_CMD. When flash is powered via MSC_CMD the MSC_IF.PWRUPF interrupt flag will be set when power up is complete and the CPU will be interrupted if MSC_IEN.PWRUPF is set.

6.3.8 Flash EM0 / EM1 Power Down

It is also possible to put the flash in a power-saving sleep mode when the system is in EM0 or EM1. Flash power down can be configured to happen on entering EM1, radio-only sleep, or with an immediate manual operation.

During EM0, software can instruct the flash to go to power down mode with the MSC_CMD.PWROFF command. Any system IRQ or flash read will wake the flash. The MSC_CMD.PWRUP command is used to power the flash back up in the absence of a wake event.

The MSC_PWRCTRL register allows the flash to be configured to automatically enter sleep mode on entering EM1 or radio-only sleep with the bits PWROFFONEM1ENTRY and PWROFFONEM1PENTRY, respectively. If the flash is configured to sleep during one of these states, it may sometimes be powered back up without processor intervention in EM0 (for example, if DMA reads flash in EM1). By default, the flash will remain powered on after such access. If the PWROFFENTRYAGAIN bit is set, it instructs the flash to re-enter the power down state if no further access is seen during the timeout period defined by PWROFFDLY. Flash must be idle for PWROFFDLY * 64 bus clocks before it will enter sleep again.

6.3.9 Wait-states

Since the CPU may be clocked faster than the flash can respond it is necessary to configure wait-states for flash accesses at higher CPU clock speeds. See the device Datasheet for information on the maximum allowed frequency for each wait-state setting. To configure the flash wait-states set the MODE field in MSC_READCTRL.

When changing wait states, care should be taken that the system is never in an invalid state. To ensure this, MODE should be changed after the clock is changed when reducing clock speed and before the clock is changed when increasing clock speed.

6.3.10 Cortex-M33 If-Then Block Folding

The Cortex-M33 offers a mechanism known as if-then block folding. This is a form of speculative prefetching where small if-then blocks are collapsed in the prefetch buffer if the condition evaluates to false. The instructions in the block then appear to execute in zero cycles. With this scheme, performance is optimized at the cost of higher energy consumption as the processor fetches more instructions from memory than it actually executes. To disable the mode, write a 1 to the DISFOLD bit in the NVIC Auxiliary Control Register; see the Cortex-M33 Technical Reference Manual for details. Normally, it is expected that this feature is most efficient when operating with 0 wait-states. Folding is enabled by default.

6.3.11 Line Buffering (Prefetch)

The MSC reads a 2 word line from flash on any flash access. The data being accessed is returned immediately and the other word locally cached so that it can be provided immediately if accessed. This has the effect of pre-fetching the second word when the first is read resulting in fewer wait-states when executing sequential code. This feature may be disabled by setting DOUTBUFEN in MSC_READCTRL.

6.3.12 Erase and Write Operations

The 20 MHz FSRCO is used for timing during flash write and erase operations. The default values in MSC_FLASHPROGRAMTIME and MSC_FLASHERASETIME contain the recommended programming configuration.

To erase a page first set WREN in MSC_WRITECTRL and load any address in the page to be erased into the MSC_ADDRB register. Next check INVADDR, LOCKED, and WREADY in MSC_STATUS to ensure that the address is valid, not locked, and the MSC is ready to modify flash. Writing ERASEPAGE in MSC_WRITEMD will execute the page erase operation. ERASE in MSC_IF will be set when the page erase is complete. If ERASE in MSC_IEN is set, the end of a page erase will also trigger an interrupt. Finally, clear WREN to disable flash operations.

In addition to a page erase, a mass erase will clear the entire contents of the main flash array. A mass erase can be initiated by the Secure Element. User Data page contents are not included in a mass erase.

To perform a programming operation, set WREN and load the address to be programmed into the MSC_ADDRB register. Next check INVADDR, LOCKED, WREADY, and WDATAREADY in MSC_STATUS to ensure that the address is valid, not locked, the MSC is ready to modify flash, and the write data buffer is clear. Writing data to MSC_WDATA will begin the programming operation. If a burst write is being performed, the next data word can be programmed to MSC_WDATA as soon as WDATAREADY is set. WRITE in MSC_IF will be set when the programming operation is complete. If WRITE in MSC_IEN is set, the end of the program operation will also trigger an interrupt. Finally, clear WREN to disable flash operations.

If data is written to the MSC_WDATA register faster than it can be processed, WDATAOV in MSC_IF will be set. If WDATAOV in MSC_IEN is set an interrupt will also be fired.

The MSC_ADDRB register only has to be written once when writing to sequential words. After each word is written, ADDRB is incremented automatically by 4. The INVADDR bit of the MSC_STATUS register is set if the loaded address is outside the flash. The LOCKED bit of the MSC_STATUS register is set if the page addressed is locked. Any attempts to erase or write to the page are ignored if INVADDR or the LOCKED bits of the MSC_STATUS register are set.

Write and erase operations may be aborted by software. To abort an erase, set the ERASEABORT bit in the MSC_WRITECMD register. To abort a write, set WRITEEND in MSC_WRITECMD

For a DMA write, CLEARWDATA in MSC_WRITECMD to assert a DMA request and transfer the first word. Alternately the first word may be programmed manually into MSC_WDATA by code.

By default, if any interrupt occurs during an erase operation, the erase is aborted. This feature may be disabled by clearing IRQERASEABORT in MSC_WRITECTRL. When an erase is aborted due to an interrupt, ERASEABORTED in MSC_STAUTS is set by hardware.

Software may observe the status of the MSC via the MSC_STATUS register. When a flash operation is in progress, BUSY will be set. If a flash operation has been requested but not yet started, PENDING will be set. This may occur if a subsystem such as the radio controller is performing MSC operations. When the write buffer underflows, TIMEOUT will be set. Buffer underflow is a normal part of the write procedure since it will occur once the last word has been written and no more data is available.

The Flash memory is organized into 64-bit wide double-words. Each 64-bit double-word can be written only twice between erase cycles. The lower and upper 32-bit words may be written sequentially in any order, or one at a time. Each flash bit is 1 after erase. Writing a 0 will clear the bit. Writing a 1 will not change the bit value.

While it is possible to write twice to the lower or upper 32-bit word of the 64-bit double word, then the other 32-bit word cannot be used. In this case, it is permitted to write to either the lower or upper 32-bit word twice between each erase, so long as no bit is ever cleared more than once.

Note: The ERASEPAGE, and CMD_WDATA registers cannot safely be written from code in Flash. It is recommended to place a small code section in RAM to set these bits and wait for the operation to complete. Also note that DMA transfers to or from any other address in Flash while a write or erase operation is in progress will produce unpredictable results.

Note: During a write or erase, flash read accesses will be stalled, effectively halting code execution from flash. Code execution continues upon write/erase completion. Code residing in RAM or ICACHE may be executed during a write/erase operation.

6.3.12.1 Low-Power Write/Erase

To limit maximum current, the programming operations can be slowed down. Set LPWRITE in MSC_WRITECTRL to double the write/erase time, halving the write/erase current.

6.3.12.2 Flash Lock

The ability to program or erase individual flash pages may be disabled using the MSC_PAGELOCKn registers. The bits in these registers may only be set to 1 by software on the device and are cleared when the device is reset. This means that once locked, a page may not be unlocked until a reset occurs. Users wishing to lock accesses to flash should implement code to write to the MSC_PAGELOCKn registers immediately after a reset. Any page locked in this way cannot be written to or erased.

Similarly, code may lock the user data page by setting UDLOCKBIT in MSC_MISCLOCKWORD, and mass erase may be disabled by setting MELOCKBIT in MSC_MISCLOCKWORD.

6.4 DEVINFO - Device Info Page

The Device Info Page holds factory programmed information about the device. It contains the following data:

- Calibration values for reconfiguring the device
- Unique ID's
- OPN identifiers (family, feature set, flash size, etc.)

6.4.1 DEVINFO Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	DEVINFO_INFO	R	DI Information
0x004	DEVINFO_PART	R	Part Info
0x008	DEVINFO_MEMINFO	R	Memory Info
0x00C	DEVINFO_MSIZE	R	Memory Size
0x010	DEVINFO_PKGINFO	R	Misc Device Info
0x014	DEVINFO_CUSTOMINFO	R	Custom Part Info
0x018	DEVINFO_SWFIX	R	SW Fix Register
0x01C	DEVINFO_SWCAPA0	R	Software Restriction
0x020	DEVINFO_SWCAPA1	R	Software Restriction
0x028	DEVINFO_EXTINFO	R	External Component Info
0x040	DEVINFO_EUI48L	R	EUI 48 Low
0x044	DEVINFO_EUI48H	R	EUI 48 High
0x048	DEVINFO_EUI64L	R	EUI64 Low
0x04C	DEVINFO_EUI64H	R	EUI64 High
0x050	DEVINFO_CALTEMP	R	Calibration temperature Information
0x054	DEVINFO_EMUTEMP	R	EMU Temperature Sensor Calibration Information
0x058	DEVINFO_HFRCODPLLCALn	R	HFRCODPLL Calibration
0x130	DEVINFO_MODULENAME0	R	Module Name Information
0x134	DEVINFO_MODULENAME1	R	Module Name Information
0x138	DEVINFO_MODULENAME2	R	Module Name Information
0x13C	DEVINFO_MODULENAME3	R	Module Name Information
0x140	DEVINFO_MODULENAME4	R	Module Name Information
0x144	DEVINFO_MODULENAME5	R	Module Name Information
0x148	DEVINFO_MODULENAME6	R	Module Name Information
0x14C	DEVINFO_MODULEINFO	R	Module Information
0x150	DEVINFO_MODXOCAL	R	Module External Oscillator Calibration Information
0x180	DEVINFO_IADC0GAIN0	R	IADC Gain Calibration
0x184	DEVINFO_IADC0GAIN1	R	IADC Gain Calibration
0x188	DEVINFO_IADC0OFFSETCAL0	R	IADC Offset Calibration
0x18C	DEVINFO_IADC0NORMALOFF-SETCAL0	R	IADC Offset Calibration
0x190	DEVINFO_IADC0NORMALOFF-SETCAL1	R	IADC Offset Calibration
0x194	DEVINFO_IADC0HISPDOFF-SETCAL0	R	IADC Offset Calibration
0x198	DEVINFO_IADC0HISPDOFF-SETCAL1	R	IADC Offset Calibration

Offset	Name	Type	Description
0x1FC	DEVINFO_LEGACY	R	Legacy Device Info
0x25C	DEVINFO_RTHERM	R	Thermistor Calibration

6.4.2 DEVINFO Register Description

6.4.2.1 DEVINFO_INFO - DI Information

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x7								0x0								0x0															
Access	R								R								R															
Name	DEVINFOREV								PRODREV								CRC															

Bit	Name	Reset	Access	Description
31:24	DEVINFOREV	0x7	R	**DI Page Version**
	DEVINFO layout revision as unsigned integer (initially 1)			
23:16	PRODREV	0x0	R	**Production Revision**
	Production revision as unsigned integer			
15:0	CRC	0x0	R	**CRC**
	CRC of DI-page (CRC-16-CCITT)			

6.4.2.2 DEVINFO_PART - Part Info

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x0								0x0						0x0															
Access			R								R						R															
Name			FAMILY								FAMILYNUM						DEVICENUM															

Bit	Name	Reset	Access	Description
31:30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29:24	FAMILY	0x0	R	**Device Family**
	Encoded portion of the Device Family			

Value	Mode	Description
0	FG	Flex Gecko
1	MG	Mighty Gecko
2	BG	Blue Gecko
5	PG	Pearl Gecko

23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:16	FAMILYNUM	0x0	R	**Device Family**
	Numeric portion of the Device Family			
15:0	DEVICENUM	0x0	R	**Device Number**
	Device Number. The device number is one letter and 3 digits. NUMBER = (alpha-'A')*1000 + numeric. 0 = A000; 1123 = B123			

6.4.2.3 DEVINFO_MEMINFO - Memory Info

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																0x0								0x0							
Access	R																R								R							
Name	DILEN																UDPAGESIZE								FLASHPAGESIZE							

Bit	Name	Reset	Access	Description
31:16	DILEN	0x0	R	**Length of DI Page**
	Length of DI area (number of 32-bit words included in CRC)			
15:8	UDPAGESIZE	0x0	R	**User Data Page Size**
	User Data page size			
7:0	FLASHPAGESIZE	0x0	R	**Flash Page Size**
	Flash page size in bytes coded as 2^((MEMINFO.PAGESIZE +10) & 0xFF. For example, the value of 0xFF = 512 bytes			

6.4.2.4 DEVINFO_MSIZE - Memory Size

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset						0x0											0x0															
Access						R											R															
Name						SRAM											FLASH															

Bit	Name	Reset	Access	Description
31:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:16	SRAM	0x0	R	**Sram Size**
	Ram size, kbyte count as unsighed integer (eg. 16)			
15:0	FLASH	0x0	R	**Flash Size**
	Flash size, kbyte count as unsigned integer (eg. 128)			

6.4.2.5 DEVINFO_PKGINFO - Misc Device Info

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0								0x0								0x0							
Access									R								R								R							
Name									PINCOUNT								PKGTYPE								TEMPGRADE							

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:16	PINCOUNT	0x0	R	**Pin Count**
	Device pin count as unsigned integer (eg. 48)			
15:8	PKGTYPE	0x0	R	**Package Type**
	Package identifier as character			

Value	Mode	Description
74	WLCSP	WLCSP package
76	BGA	BGA package
77	QFN	QFN package
81	QFP	QFP package

Bit	Name	Reset	Access	Description
7:0	TEMPGRADE	0x0	R	**Temperature Grade**
	Temperature Grade of produt as unsigned integer enumeration			

Value	Mode	Description
0	N40TO85	-40 to 85 degC
1	N40TO125	-40 to 125 degC
2	N40TO105	-40 to 105 degC
3	N0TO70	0 to 70 degC

6.4.2.6 DEVINFO_CUSTOMINFO - Custom Part Info

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	PARTNO																															

Bit	Name	Reset	Access	Description
31:16	PARTNO	0x0	R	**Part Number**
	Custom part identifier as unsigned integer (eg. 903). 65535 for standard product			
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

6.4.2.7 DEVINFO_SWFIX - SW Fix Register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFFFFFFFF																															
Access	R																															
Name	RSV																															

Bit	Name	Reset	Access	Description
31:0	RSV	0xFFFFFFFF	R	**Reserved**
	Reserved for future use			

6.4.2.8 DEVINFO_SWCAPA0 - Software Restriction

Offset	Bit Position																											
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x0				0x0				0x0				0x0				0x0				0x0	
Access											R				R				R				R				R				R	
Name											SRI				CONNECT				BTSMART				RF4CE				THREAD				ZIGBEE	

Bit	Name	Reset	Access	Description
31:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	SRI	0x0	R	**RAIL Capability**
	RAIL capability level			
	Value	Mode		Description
	0	LEVEL0		RAIL capability not available
	1	LEVEL1		RAIL enabled
	2	LEVEL2		N/A
	3	LEVEL3		N/A
19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	CONNECT	0x0	R	**Connect Capability**
	Connect stack capability level			
	Value	Mode		Description
	0	LEVEL0		Connect stack capability not available
	1	LEVEL1		Connect enabled
	2	LEVEL2		N/A
	3	LEVEL3		N/A
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	BTSMART	0x0	R	**Bluetooth Smart Capability**
	Bluetooth SMART stack capability level			
	Value	Mode		Description
	0	LEVEL0		Bluetooth SMART stack capability not available
	1	LEVEL1		Bluetooth SMART enabled
	2	LEVEL2		N/A
	3	LEVEL3		N/A

Bit	Name	Reset	Access	Description
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	RF4CE	0x0	R	**RF4CE Capability**
	RF4CE stack capability level			
	Value	Mode		Description
	0	LEVEL0		RF4CE stack capability not available
	1	LEVEL1		RF4CE stack enabled
	2	LEVEL2		N/A
	3	LEVEL3		N/A
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	THREAD	0x0	R	**Thread Capability**
	Thread stack capability level			
	Value	Mode		Description
	0	LEVEL0		Thread stack capability not available
	1	LEVEL1		Thread stack enabled
	2	LEVEL2		N/A
	3	LEVEL3		N/A
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	ZIGBEE	0x0	R	**Zigbee Capability**
	ZigBee stack capability level			
	Value	Mode		Description
	0	LEVEL0		Zigbee stack capability not available
	1	LEVEL1		Green Power only
	2	LEVEL2		Zigbee and Green Power
	3	LEVEL3		Zigbee Only

6.4.2.9 DEVINFO_SWCAPA1 - Software Restriction

Offset	Bit Position																														
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														R	R	R
Name																														GWEN	NCPEN	RFMCUEN

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	GWEN	0x0	R	**Gateway**
	Gateway enabled part			
1	NCPEN	0x0	R	**NCP**
	Network co-processor enabled part. NCP only if RFMCUEN = 0			
0	RFMCUEN	0x0	R	**RF-MCU**
	RF-MCU enabled part. RF-MCU only if NCPEN = 0			

6.4.2.10 DEVINFO_EXTINFO - External Component Info

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0								0x0								0x0							
Access									R								R								R							
Name									REV								CONNECTION								TYPE							

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:16	REV	0x0	R	**Revision**
	MCM Revision			
15:8	CONNECTION	0x0	R	**Connection**
	Connection protocol to external interface			
	Value	Mode		Description
	0	SPI		SPI control interface
	255	NONE		No interface
7:0	TYPE	0x0	R	**Type**
	External Component			
	Value	Mode		Description
	255	NONE		

6.4.2.11 DEVINFO_EUI48L - EUI 48 Low

Offset	Bit Position																													
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0								0x0																							
Access	R								R																							
Name	OUI48L								UNIQUEID																							

Bit	Name	Reset	Access	Description
31:24	OUI48L	0x0	R	**OUI48L**
	Lower Octet of EUI48 Organizationally Unique Identifier			
23:0	UNIQUEID	0x0	R	**Unique ID**
	Unique identifier			

6.4.2.12 DEVINFO_EUI48H - EUI 48 High

Offset	Bit Position																													
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFFFF																0x0															
Access	R																R															
Name	RESERVED																OUI48H															

Bit	Name	Reset	Access	Description
31:16	RESERVED	0xFFFF	R	**RESERVED**
15:0	OUI48H	0x0	R	**OUI48H**
	Upper two Octets of EUI48 OUI			

6.4.2.13 DEVINFO_EUI64L - EUI64 Low

Offset	Bit Position																													
0x048	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	UNIQUEL																															

Bit	Name	Reset	Access	Description
31:0	UNIQUEL	0x0	R	**UNIQUEL**
	Lower 32 bits of EUI64 Unique Identifier			

6.4.2.14 DEVINFO_EUI64H - EUI64 High

Offset	Bit Position																													
0x04C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																								0x0							
Access	R																								R							
Name	OUI64																								UNIQUEH							

Bit	Name	Reset	Access	Description
31:8	OUI64	0x0	R	**OUI64**
	24-bit OUI identifier			
7:0	UNIQUEH	0x0	R	**UNIQUEH**
	Upper 8 bits of EUI64 unique identifier			

6.4.2.15 DEVINFO_CALTEMP - Calibration temperature Information

Offset	Bit Position																													
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									TEMP							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	TEMP	0x0	R	**Cal Temp**
	Calibration temperature as an unsigned int in DegC. (0x19 = 25 DegC)			

6.4.2.16 DEVINFO_EMUTEMP - EMU Temperature Sensor Calibration Information

Offset	Bit Position																													
0x054	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x0										
Access																						R										
Name																						EMUTEMPROOM										

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:2	EMUTEMPROOM	0x0	R	**Emu Room Temperature**
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

6.4.2.17 DEVINFO_HFRCODPLLCALn - HFRCODPLL Calibration

Offset	Bit Position																													
0x058	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0				0x0		0x0		0x0			0x0					0x0		0x0							0x0						
Access	R				R		R		R			R					R		R							R						
Name	IREFTC				CMPSEL		CLKDIV		CMPBIAS			FREQRANGE					LDOHP		FINETUNING							TUNING						

Bit	Name	Reset	Access	Description
31:28	IREFTC	0x0	R	
	Tempco Trim			
27:26	CMPSEL	0x0	R	
	Comparator Load Select			
25:24	CLKDIV	0x0	R	
	Locally Divide HFRCO Clock Output			
23:21	CMPBIAS	0x0	R	
	Comparator Bias Current			
20:16	FREQRANGE	0x0	R	
	Frequency Range			
15	LDOHP	0x0	R	
	LDO High Power Mode			
14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	FINETUNING	0x0	R	
	Fine Tuning Value			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	TUNING	0x0	R	
	Tuning Value			

6.4.2.18 DEVINFO_MODULENAME0 - Module Name Information

Offset	Bit Position																													
0x130	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFF								0xFF								0xFF								0xFF							
Access	R								R								R								R							
Name	MODCHAR4								MODCHAR3								MODCHAR2								MODCHAR1							

Bit	Name	Reset	Access	Description
31:24	MODCHAR4	0xFF	R	
	Fourth character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
23:16	MODCHAR3	0xFF	R	
	Third character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
15:8	MODCHAR2	0xFF	R	
	Second character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
7:0	MODCHAR1	0xFF	R	
	First character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			

6.4.2.19 DEVINFO_MODULENAME1 - Module Name Information

Offset	Bit Position																													
0x134	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFF								0xFF								0xFF								0xFF							
Access	R								R								R								R							
Name	MODCHAR8								MODCHAR7								MODCHAR6								MODCHAR5							

Bit	Name	Reset	Access	Description
31:24	MODCHAR8	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
23:16	MODCHAR7	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
15:8	MODCHAR6	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
7:0	MODCHAR5	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			

6.4.2.20 DEVINFO_MODULENAME2 - Module Name Information

Offset	Bit Position																													
0x138	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFF								0xFF								0xFF								0xFF							
Access	R								R								R								R							
Name	MODCHAR12								MODCHAR11								MODCHAR10								MODCHAR9							

Bit	Name	Reset	Access	Description
31:24	MODCHAR12	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
23:16	MODCHAR11	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
15:8	MODCHAR10	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
7:0	MODCHAR9	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			

6.4.2.21 DEVINFO_MODULENAME3 - Module Name Information

Offset	Bit Position																													
0x13C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFF								0xFF								0xFF								0xFF							
Access	R								R								R								R							
Name	MODCHAR16								MODCHAR15								MODCHAR14								MODCHAR13							

Bit	Name	Reset	Access	Description
31:24	MODCHAR16	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
23:16	MODCHAR15	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
15:8	MODCHAR14	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
7:0	MODCHAR13	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			

6.4.2.22 DEVINFO_MODULENAME4 - Module Name Information

Offset	Bit Position																													
0x140	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFF								0xFF								0xFF								0xFF							
Access	R								R								R								R							
Name	MODCHAR20								MODCHAR19								MODCHAR18								MODCHAR17							

Bit	Name	Reset	Access	Description
31:24	MODCHAR20	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
23:16	MODCHAR19	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
15:8	MODCHAR18	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
7:0	MODCHAR17	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			

6.4.2.23 DEVINFO_MODULENAME5 - Module Name Information

Offset	Bit Position																													
0x144	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFF								0xFF								0xFF								0xFF							
Access	R								R								R								R							
Name	MODCHAR24								MODCHAR23								MODCHAR22								MODCHAR21							

Bit	Name	Reset	Access	Description
31:24	MODCHAR24	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
23:16	MODCHAR23	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
15:8	MODCHAR22	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			
7:0	MODCHAR21	0xFF	R	
	Character of Module Name, 0xFF = unwritten, 0x00 = character not used in name			

6.4.2.24 DEVINFO_MODULENAME6 - Module Name Information

Offset	Bit Position																													
0x148	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFFFF																0xFF								0xFF							
Access	R																R								R							
Name	RSV																MODCHAR26								MODCHAR25							

Bit	Name	Reset	Access	Description
31:16	RSV	0xFFFF	R	
	Reserved for future use			
15:8	MODCHAR26	0xFF	R	
	Last possible character of module name, 0xFF = unwritten, 0x00 = character not used in name			
7:0	MODCHAR25	0xFF	R	
	0xFF = unwritten, 0x00 = character not used in name			

6.4.2.25 DEVINFO_MODULEINFO - Module Information

Offset	Bit Position																													
0x14C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1	0x1	0x1	0x1FF									0x1	0x1	0x1	0x1	0x1	0x7F							0x7			0x1F				
Access	R	R	R	R									R	R	R	R	R	R							R			R				
Name	EXTVALID	PHYLIMITED	PADCDC	MODNUMBERMSB									HFXOCALVAL	LFXOCALVAL	EXPRESS	LFXO	TYPE	MODNUMBER							ANTENNA			HWREV				

Bit	Name	Reset	Access	Description
31	EXTVALID	0x1	R	
	EXTINFO entry used			
	Value	Mode		Description
	0	EXTUSED		EXT used
	1	EXTUNUSED		EXT not used
30	PHYLIMITED	0x1	R	
	PHY Limited			
	Value	Mode		Description
	0	LIMITED		
	1	UNLIMITED		
29	PADCDC	0x1	R	
	PAVDD Connection			
	Value	Mode		Description
	0	VDCDC		PAVDD connected to Vdcdc
	1	OTHER		PAVDD connected to Vdd or other
28:20	MODNUMBERMSB	0x1FF	R	
	Counter allowing unique identification of module per lookup when combined with MODNUMBER			
19	HFXOCALVAL	0x1	R	
	HFXO Factory Calibrated			
	Value	Mode		Description
	0	VALID		HFXO calibration in MODXOCAL is valid
	1	NOTVALID		HFXO calibration in MODXOCAL is not valid
18	LFXOCALVAL	0x1	R	

Bit	Name	Reset	Access	Description
	LFXO Factory Calibrated			
	Value	Mode		Description
	0	VALID		LFXO Tuning in MODXOCAL is valid
	1	NOTVALID		LFXO Tuning value in MODXOCAL is not valid
17	EXPRESS	0x1	R	
	Blue Gecko Express			
	Value	Mode		Description
	0	SUPPORTED		Blue Gecko Express is supported
	1	NONE		Blue Gecko Express is not supported
16	LFXO	0x1	R	
	Module has LFXO			
	Value	Mode		Description
	0	NONE		LFXO is not installed
	1	PRESENT		LFXO is installed
15	TYPE	0x1	R	
	Module Type			
	Value	Mode		Description
	0	PCB		PCB
	1	SIP		SIP
14:8	MODNUMBER	0x7F	R	
	Counter allowing unique identification of module per lookup when combined with MODNUMBER MSB			
7:5	ANTENNA	0x7	R	
	Module Antenna Type			
	Value	Mode		Description
	0	BUILTIN		Built-in Antenna
	1	CONNECTOR		RF Connector
	2	RFPAD		RF Pad
	3	INVERTEDF		F-invert PCB
4:0	HWREV	0x1F	R	
	Module Hardware Revision. Starting from 0			

6.4.2.26 DEVINFO_MODXOCAL - Module External Oscillator Calibration Information

Offset	Bit Position																													
0x150	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset										0x7F							0xFF								0xFF							
Access										R							R								R							
Name										LFXOCAPTUNE							HFXOCTUNEXOANA								HFXOCTUNEXIANA							

Bit	Name	Reset	Access	Description
31:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:16	LFXOCAPTUNE	0x7F	R	
	LFXO Cap Tuning			
15:8	HFXOCTUNEXOANA	0xFF	R	
	Tuning capacitance on XO			
7:0	HFXOCTUNEXIANA	0xFF	R	
	Tuning capacitance on XI			

6.4.2.27 DEVINFO_IADC0GAIN0 - IADC Gain Calibration

Offset	Bit Position																													
0x180	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x0																	0x0							
Access								R																	R							
Name								GAINCANA2																	GAINCANA1							

Bit	Name	Reset	Access	Description
31:16	GAINCANA2	0x0	R	
	Input Gain = 2x			
15:0	GAINCANA1	0x0	R	
	Input Gain = 1x and 0.5x			

6.4.2.28 DEVINFO_IADC0GAIN1 - IADC Gain Calibration

Offset	Bit Position																													
0x184	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																0x0															
Access	R																R															
Name	GAINCANA4																GAINCANA3															

Bit	Name	Reset	Access	Description
31:16	GAINCANA4	0x0	R	
	Input Gain = 4x			
15:0	GAINCANA3	0x0	R	
	Input Gain = 3x			

6.4.2.29 DEVINFO_IADC0OFFSETCAL0 - IADC Offset Calibration

Offset	Bit Position																													
0x188	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																0x0															
Access	R																R															
Name	OFFSETANA1HIACC																OFFSETANABASE															

Bit	Name	Reset	Access	Description
31:16	OFFSETANA1HIACC	0x0	R	
	High-accuracy OSR adjustment term			
15:0	OFFSETANABASE	0x0	R	
	Base analog offset term			

6.4.2.30 DEVINFO_IADC0NORMALOFFSETCAL0 - IADC Offset Calibration

Offset	Bit Position																													
0x18C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																0x0															
Access	R																R															
Name	OFFSETANA2NORM																OFFSETANA1NORM															

Bit	Name	Reset	Access	Description
31:16	OFFSETANA2NORM	0x0	R	
	Normal mode offset gain adjustment term			
15:0	OFFSETANA1NORM	0x0	R	
	Normal mode analog offset term at OSR=2x, gain = 1x			

6.4.2.31 DEVINFO_IADC0NORMALOFFSETCAL1 - IADC Offset Calibration

Offset	Bit Position																													
0x190	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	R															
Name																	OFFSETANA3NORM															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	OFFSETANA3NORM	0x0	R	
	Normal mode offset term for OSR>=4x			

6.4.2.32 DEVINFO_IADC0HISPDOFFSETCAL0 - IADC Offset Calibration

Offset	Bit Position																													
0x194	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																0x0															
Access	R																R															
Name	OFFSETANA2HISPD																OFFSETANA1HISPD															

Bit	Name	Reset	Access	Description
31:16	OFFSETANA2HISPD	0x0	R	
	High speed mode offset gain adjustment term			
15:0	OFFSETANA1HISPD	0x0	R	
	High speed mode analog offset term at OSR=2x, gain = 1x			

6.4.2.33 DEVINFO_IADC0HISPDOFFSETCAL1 - IADC Offset Calibration

Offset	Bit Position																													
0x198	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	R															
Name																	OFFSETANA3HISPD															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	OFFSETANA3HISPD	0x0	R	
	High-speed mode offset term for OSR>=4x			

6.4.2.34 DEVINFO_LEGACY - Legacy Device Info

Offset	Bit Position																													
0x1FC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x80																				
Access												R																				
Name												DEVICEFAMILY																				

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:16	DEVICEFAMILY Device Family	0x80	R	**Device Family**

Value	Mode	Description
16	EFR32MG1P	EFR32 Mighty Gecko Family Series 1 Device Config 1
17	EFR32MG1B	EFR32 Mighty Gecko Family Series 1 Device Config 1
18	EFR32MG1V	EFR32 Mighty Gecko Family Series 1 Device Config 1
19	EFR32BG1P	EFR32 Blue Gecko Family Series 1 Device Config 1
20	EFR32BG1B	EFR32 Blue Gecko Family Series 1 Device Config 1
21	EFR32BG1V	EFR32 Blue Gecko Family Series 1 Device Config 1
25	EFR32FG1P	EFR32 Flex Gecko Family Series 1 Device Config 1
26	EFR32FG1B	EFR32 Flex Gecko Family Series 1 Device Config 1
27	EFR32FG1V	EFR32 Flex Gecko Family Series 1 Device Config 1
28	EFR32MG12P	EFR32 Mighty Gecko Family Series 1 Device Config 2
29	EFR32MG12B	EFR32 Mighty Gecko Family Series 1 Device Config 2
30	EFR32MG12V	EFR32 Mighty Gecko Family Series 1 Device Config 2
31	EFR32BG12P	EFR32 Blue Gecko Family Series 1 Device Config 2
32	EFR32BG12B	EFR32 Blue Gecko Family Series 1 Device Config 2
33	EFR32BG12V	EFR32 Blue Gecko Family Series 1 Device Config 2
37	EFR32FG12P	EFR32 Flex Gecko Family Series 1 Device Config 2
38	EFR32FG12B	EFR32 Flex Gecko Family Series 1 Device Config 2
39	EFR32FG12V	EFR32 Flex Gecko Family Series 1 Device Config 2
40	EFR32MG13P	EFR32 Mighty Gecko Family Series 13 Device Config 3
41	EFR32MG13B	EFR32 Mighty Gecko Family Series 13 Device Config 3
42	EFR32MG13V	EFR32 Mighty Gecko Family Series 1 Device Config 3
43	EFR32BG13P	EFR32 Blue Gecko Family Series 1 Device Config 3

Bit	Name	Reset	Access	Description
	44	EFR32BG13B		EFR32 Blue Gecko Family Series 1 Device Config 3
	45	EFR32BG13V		EFR32 Blue Gecko Family Series 1 Device Config 3
	49	EFR32FG13P		EFR32 Flex Gecko Family Series 1 Device Config 3
	50	EFR32FG13B		EFR32 Flex Gecko Family Series 1 Device Config 3
	51	EFR32FG13V		EFR32 Flex Gecko Family Series 1 Device Config 3
	52	EFR32MG14P		EFR32 Mighty Gecko Family Series 1 Device Config 4
	53	EFR32MG14B		EFR32 Mighty Gecko Family Series 1 Device Config 4
	54	EFR32MG14V		EFR32 Mighty Gecko Family Series 1 Device Config 4
	55	EFR32BG14P		EFR32 Blue Gecko Family Series 1 Device Config 4
	56	EFR32BG14B		EFR32 Blue Gecko Family Series 1 Device Config 4
	57	EFR32BG14V		EFR32 Blue Gecko Family Series 1 Device Config 4
	61	EFR32FG14P		EFR32 Flex Gecko Family Series 1 Device Config 4
	62	EFR32FG14B		EFR32 Flex Gecko Family Series 1 Device Config 4
	63	EFR32FG14V		EFR32 Flex Gecko Family Series 1 Device Config 4
	71	EFM32G		EFM32 Gecko Device Family
	72	EFM32GG		EFM32 Giant Gecko Device Family
	73	EFM32TG		EFM32 Tiny Gecko Device Family
	74	EFM32LG		EFM32 Leopard Gecko Device Family
	75	EFM32WG		EFM32 Wonder Gecko Device Family
	76	EFM32ZG		EFM32 Zero Gecko Device Family
	77	EFM32HG		EFM32 Happy Gecko Device Family
	81	EFM32PG1B		EFM32 Pearl Gecko Device Family Series 1 Device Config 1
	83	EFM32JG1B		EFM32 Jade Gecko Device Family Series 1 Device Config 1
	85	EFM32PG12B		EFM32 Pearl Gecko Device Family Series 1 Device Config 2
	87	EFM32JG12B		EFM32 Jade Gecko Device Family Series 1 Device Config 2
	89	EFM32PG13B		EFM32 Pearl Gecko Device Family Series 1 Device Config 3
	91	EFM32JG13B		EFM32 Jade Gecko Device Family Series 1 Device Config 3
	100	EFM32GG11B		EFM32 Giant Gecko Device Family Series 1 Device Config 1
	103	EFM32TG11B		EFM32 Giant Gecko Device Family Series 1 Device Config 1
	120	EZR32LG		EZR32 Leopard Gecko Device Family
	121	EZR32WG		EZR32 Wonder Gecko Device Family
	122	EZR32HG		EZR32 Happy Gecko Device Family
	128	SERIES2V0		DI page is encoded with the series 2 layout. Check alternate location.
15:0	*Reserved*			*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*

6.4.2.35 DEVINFO_RTHERM - Thermistor Calibration

Offset	Bit Position																													
0x25C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									RTHERM							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	RTHERM	0x0	R	
	Calibrated Thermistor Resistor			

6.5 ICACHE - Instruction Cache

The ICACHE provides fast access to recently executed instructions improving both speed and power consumption of code execution. The instruction cache is enabled by default, but can be disabled by setting CACHEDIS in ICACHE_CTRL. When enabled, the instruction cache typically reduces the number of flash reads significantly, thus saving energy. In most cases, a cache hit-rate of more than 70 % is achievable. When a 32-bit instruction fetch hits in the cache, the data is returned to the processor in one clock cycle, bypassing the flash accesses wait-states. The cache content is retained in EM2 and EM3.

The instruction cache is connected directly to the CODE bus on the ARM core and functions as a memory access filter between the processor and the memory system, as illustrated in Figure 6.1 Instruction Cache Block Diagram on page 83. The cache consists of an access filter, lookup logic, SRAM, and three performance counters. The access filter checks if a transfer is an instruction fetch located in a cacheable region. If it is the cache lookup logic and SRAM is enabled. Otherwise, the cache is bypassed and the access is forwarded to the memory system. If lookup is enabled data is either returned from the cache (hit) or fetch from the memory system and cached (miss).

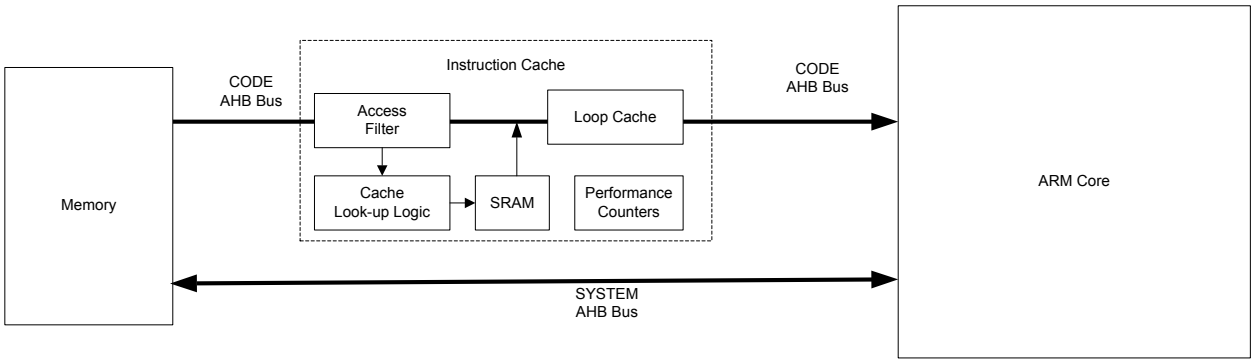


Figure 6.1. Instruction Cache Block Diagram

Note that while all access to code spaces use the CODE bus only instruction fetches are cached. Data accesses to the CODE region are passed through the ICACHE.

6.5.1 Cache Operation

It is highly recommended to keep the cache enabled. To improve cache-efficiency, sections of code with very low cache hit rate should not be cached. This is achieved by placing these code sections in non-cacheable MPU regions and setting USEMPU in ICACHE_CTRL. When USEMPU is set, instruction fetches to non-cacheable MPU regions will not be looked up or saved in cache. This feature may also be used to avoid instructions from low-power memory taking up space from more power-hungry memory. For more information on the MPU see the ARM Cortex-M33 MPU documentation.

The optional loop-cache is optimized to store smaller code-loops efficiently. The loop-cache is enabled when LPLEVEL in ICACHE_LPMODE is set to ADVANCED or MINACTIVITY. The difference between the two settings is that when MINACTIVITY is selected loop-cache outputs may be gated off to reduce power at the cost of more wait-states due to loop-cache misses. Having LPLEVEL set to BASIC disables the loop-cache functionality completely. NESTFACTOR in ICACHE _LPMODE is used to decide when to stick with the currently detected loop rather than start tracking a new loop. Optimal value will depend on the actual code running, meaning that this setting may be tuned for optimal performance.

By default, the instruction cache is automatically invalidated when the contents of the flash is changed (i.e. written or erased). In many cases, however, the application only makes changes to data in the flash, not code. In this case, the automatic invalidate feature can be disabled by setting AUTOFLUSHDIS in ICACHE_CTRL. The cache can also be manually invalidated by writing 1 to FLUSH in ICACHE_CMD.

In the event that a parity error in the cache is detected, the RAMERROR flag will be set in ICACHE_IF. The data is automatically reloaded when this occurs so no action is required by software. This flag informational only, and can be used to detect the rate of corruption events. If RAMERROR in ICACHE_IEN is set, an interrupt will be triggered.

The cache is automatically flushed whenever a BUS-FAULT occurs. If this occurs during performance counting the counts will be effected.

6.5.2 Performance Measurement

To measure the hit-rate of a code-section, the built-in performance counters can be used. Before the section, start the performance counters by setting STARTPC in ICACHE_CMD register. This starts the performance counters, counting from 0. At the end of the section, stop the performance counters by setting STOPPC in ICACHE_CMD. The number of cache hits and cache misses for that section can then be read from PCHITS and PCMISSES. The cache hit-ratio can be calculated as PCHITS / (PCHITS + PCMISSES). PCAHITS contains the loopcache hits only. Any hits in PCAHITS are also counted in PCHITS. The loopcache hit-ratio can be calculated as PCAHITS / (PCHITS + PCMISSES). When PCHITS/PCAHITS/PCMISSES overflow, the HITOF/AHITOF/MISSOF interrupt flags are set respectively. These flags must be cleared by software. The range of the performance counters can be extended by increasing a counter in the interrupt routine. The performance counters only count when a cache lookup is performed. Access to non-cacheable regions, data fetches, and access made while the ICACHE is disabled do not increment PCMISSES.

Software may check the if the performance counters are running using PCRUNNING in ICACHE_STATUS.

6.5.3 ICACHE Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	ICACHE_IPVERSION	R	IP Version
0x004	ICACHE_CTRL	RW	Control Register
0x008	ICACHE_PCHITS	RH	Performance Counter Hits
0x00C	ICACHE_PCMISSES	RH	Performance Counter Misses
0x010	ICACHE_PCAHITS	RH	Performance Counter Advanced Hits
0x014	ICACHE_STATUS	RH	Status Register
0x018	ICACHE_CMD	W	Command Register
0x01C	ICACHE_LPMODE	RW	Low Power Mode
0x020	ICACHE_IF	RWH INTFLAG	Interrupt Flag
0x024	ICACHE_IEN	RW	Interrupt Enable
0x1000	ICACHE_IPVERSION_SET	R	IP Version
0x1004	ICACHE_CTRL_SET	RW	Control Register
0x1008	ICACHE_PCHITS_SET	RH	Performance Counter Hits
0x100C	ICACHE_PCMISSES_SET	RH	Performance Counter Misses
0x1010	ICACHE_PCAHITS_SET	RH	Performance Counter Advanced Hits
0x1014	ICACHE_STATUS_SET	RH	Status Register
0x1018	ICACHE_CMD_SET	W	Command Register
0x101C	ICACHE_LPMODE_SET	RW	Low Power Mode
0x1020	ICACHE_IF_SET	RWH INTFLAG	Interrupt Flag
0x1024	ICACHE_IEN_SET	RW	Interrupt Enable
0x2000	ICACHE_IPVERSION_CLR	R	IP Version
0x2004	ICACHE_CTRL_CLR	RW	Control Register
0x2008	ICACHE_PCHITS_CLR	RH	Performance Counter Hits
0x200C	ICACHE_PCMISSES_CLR	RH	Performance Counter Misses
0x2010	ICACHE_PCAHITS_CLR	RH	Performance Counter Advanced Hits
0x2014	ICACHE_STATUS_CLR	RH	Status Register
0x2018	ICACHE_CMD_CLR	W	Command Register
0x201C	ICACHE_LPMODE_CLR	RW	Low Power Mode
0x2020	ICACHE_IF_CLR	RWH INTFLAG	Interrupt Flag
0x2024	ICACHE_IEN_CLR	RW	Interrupt Enable
0x3000	ICACHE_IPVERSION_TGL	R	IP Version
0x3004	ICACHE_CTRL_TGL	RW	Control Register
0x3008	ICACHE_PCHITS_TGL	RH	Performance Counter Hits
0x300C	ICACHE_PCMISSES_TGL	RH	Performance Counter Misses
0x3010	ICACHE_PCAHITS_TGL	RH	Performance Counter Advanced Hits

Offset	Name	Type	Description
0x3014	ICACHE_STATUS_TGL	RH	Status Register
0x3018	ICACHE_CMD_TGL	W	Command Register
0x301C	ICACHE_LPMODE_TGL	RW	Low Power Mode
0x3020	ICACHE_IF_TGL	RWH INTFLAG	Interrupt Flag
0x3024	ICACHE_IEN_TGL	RW	Interrupt Enable

6.5.4 ICACHE Register Description

6.5.4.1 ICACHE_IPVERSION - IP Version

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

6.5.4.2 ICACHE_CTRL - Control Register

Offset	Bit Position																														
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														RW	RW	RW
Name																														AUTOFLUSHDIS	USEMPU	CACHEDIS

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	AUTOFLUSHDIS	0x0	RW	**Automatic Flushing Disable**
	Disables automatic flushing based on Internal Flash write/erase			
1	USEMPU	0x0	RW	**Use MPU**
	Use MPU to select non/cacheable regions			
0	CACHEDIS	0x0	RW	**Cache Disable**
	Disables caching for all regions			

6.5.4.3 ICACHE_PCHITS - Performance Counter Hits

Offset	Bit Position																														
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	PCHITS																															

Bit	Name	Reset	Access	Description
31:0	PCHITS	0x0	R	**Performance Counter Hits**
	Hit counter value			

6.5.4.4 ICACHE_PCMISSES - Performance Counter Misses

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	PCMISSES																															

Bit	Name	Reset	Access	Description
31:0	PCMISSES	0x0	R	**Performance Counter Misses**
	Miss counter value			

6.5.4.5 ICACHE_PCAHITS - Performance Counter Advanced Hits

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	PCAHITS																															

Bit	Name	Reset	Access	Description
31:0	PCAHITS	0x0	R	**Performance Counter Advanced Hits**
	Hit counter value for hits due to Advanced Buffering mode. These hits are also represented in PCHITS.			

6.5.4.6 ICACHE_STATUS - Status Register

Offset	Bit Position																														
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																R
Name																																PCRUNNING

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	PCRUNNING	0x0	R	**PC Running**
	Performance Counters are running			

6.5.4.7 ICACHE_CMD - Command Register

Offset	Bit Position																														
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														W	W	W
Name																														STOPPC	STARTPC	FLUSH

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	STOPPC	0x0	W	**Stop Performance Counters**
	Stops the Performance Counters			
1	STARTPC	0x0	W	**Start Performance Counters**
	Starts the Performance Counters			
0	FLUSH	0x0	W	**Flush**
	Clears Cached Data			

6.5.4.8 ICACHE_LPMODE - Low Power Mode

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x2						0x3	
Access																									RW						RW	
Name																									NESTFACTOR						LPLEVEL	

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:4	NESTFACTOR	0x2	RW	**Low Power Nest Factor** Parameter used in the advanced buffering mode to control its estimation when a branch access is likely to be accssed in the near future. In general, a higher number will improve performance in code with deeply nested loops.
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	LPLEVEL	0x3	RW	**Low Power Level** Controls the low-power level of the cache. In general, the default setting is best for most applications.

Value	Mode	Description
0	BASIC	Base instruction cache functionality
1	ADVANCED	Advanced buffering mode, where the cache uses the fetch pattern to predict highly accessed data and store it in low-energy memory
3	MINACTIVITY	Minimum activity mode, which allows the cache to minimize activity in logic that it predicts has a low probability being used. This mode can introduce wait-states into the instruction fetch stream when the cache exits one of its low-activity states. The number of wait-states introduced is small, but users running with 0-wait-state memory and wishing to reduce the variability that the cache might introduce with additional wait-states may wish to lower the cache low-power level. Note, this mode includes the advanced buffering mode functionality.

6.5.4.9 ICACHE_IF - Interrupt Flag

Offset	Bit Position																														
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0						0x0	0x0	0x0
Access																								RW						RW	RW	RW
Name																								RAMERROR						AHITOF	MISSOF	HITOF

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	RAMERROR	0x0	RW	**RAM error Interrupt Flag**
	RAM parity error detected			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	AHITOF	0x0	RW	**Advanced Hit Overflow Interrupt Flag**
	Advanced hit performance counter has overflowed			
1	MISSOF	0x0	RW	**Miss Overflow Interrupt Flag**
	Miss performance counter has overflowed			
0	HITOF	0x0	RW	**Hit Overflow Interrupt Flag**
	Hit performance counter has overflowed			

6.5.4.10 ICACHE_IEN - Interrupt Enable

Offset	Bit Position																														
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0						0x0	0x0	0x0
Access																								RW						RW	RW	RW
Name																								RAMERROR						AHITOF	MISSOF	HITOF

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	RAMERROR	0x0	RW	**RAM error Interrupt Enable**
	Enable RAMERROR interrupt			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	AHITOF	0x0	RW	**Advanced Hit Overflow Interrupt Enable**
	Enable AHITOF interrupt			
1	MISSOF	0x0	RW	**Miss Overflow Interrupt Enable**
	Enable MISSOF interrupt			
0	HITOF	0x0	RW	**Hit Overflow Interrupt Enable**
	Enable HITOF interrupt			

6.6 SYSCFG - System Configuration

The SYSCFG block is used to configure SRAM. It also contains some interrupt flags for software use. The system has the following major SRAM blocks:

- DMEM0 - Primary system data memory (RAM)
- FRCRAM - Frame Rate Controller SRAM
- SEQRAM - Sequencer SRAM
- DEMODRAM - Demodulator SRAM

6.6.1 Ram Retention

DMEM0 is broken into two 24 KB and 8 KB banks, beginning at address 0x20000000 and 0x20006000, respectively. By default both banks are retained in EM2/EM3. Sleep mode current can be significantly reduced by powering down a bank that does not need to be retained. RAMRETNCTRL in the SYSCFG_DMEM0RETNCTRL register controls which banks are retained in EM2/EM3.

Note: Root code requires some RAM storage during a system reset. The RAM used by root code is located at the top of the DRAM0 memory space. If the user application requires RAM that persists through reset, it is recommended to use a statically allocated section at the bottom of the SRAM memory space (address 0x20000000).

FRCRAM and all or part of SEQRAM may be powered down in EM2/EM3 if not required. To control retention of these areas, set FRCRAMRETNCTRL or SEQRAMRETNCTRL in SYSCFG_RADIORAMRETCTRL to the desired value.

6.6.2 ECC

DMEM0, FRCRAM, and SEQRAM support one bit correction and two bit detection ECC. To enable error detection for DMEM0, set RAMECCCHKEN in SYSCFG_DMEM0ECCCTRL. To enable error detection for FRCRAM and SEQRAM, set FRCRAMECCCHKEN and SEQRAMECCCHKEN in SYSCFG_RADIOECCCTRL. To enable auto-correction of one bit errors in DMEM0, set RAMECCEWEN in SYSCFG_DMEM0ECCCTRL. To enable auto-correction of one bit errors in FRCRAM and SEQRAM, set FRCRAMECCEWEN and SEQRAMECCEWEN in SYSCFG_RADIOECCCTRL.

When ECC error events are detected, the corresponding flags in SYSCFG_IF are set. When a flag is set, an interrupt will be triggered if the corresponding interrupt enable bit is set in SYSCFG_IEN.

When an error occurs, the address of the detected error is written to SYSCFG_DMEM0ECCADDR, SYSCFG_FRCRAMECCADDR, or SYSCF_SEQRAMECCERR depending on the source of the error.

The recommend procedure for initializing ECC RAM is to first enable ECC, then write zeros to all locations. This will clear the RAM and initialize the syndrome. If the ECC RAM is not written as described, then any reads to uninitialized RAM locations will result in an ECC error.

Note: The RAM ECC feature must be enabled to achieve good long term reliability. The long term reliability of the RAM is only specified with ECC enabled.

6.6.3 Software Interrupts

The SYSCFG block also provides some software interrupts that can be used to communicate between software tasks. To trigger a software interrupt set the corresponding bit in SYSCFG_IF.

6.6.4 Bus faults

By default, two bit ECC errors and reads to unmapped addresses trigger a BusFault. These bus fault sources can be disabled by clearing RAMECCERRFAULTEN and ADDRFAULTEN in SYSCFG_CTRL.

6.6.5 SYSCFG Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	SYSCFG_IF	RWH INTFLAG	Interrupt Flag
0x004	SYSCFG_IEN	RW	Interrupt Enable
0x010	SYSCFG_CHIPREVHW	RWH	Chip Revision, Hard-wired
0x014	SYSCFG_CHIPREV	RW	Part Family and Revision Values
0x200	SYSCFG_CTRL	RW	Memory System Control
0x208	SYSCFG_DMEM0RETNCTRL	RW	DMEM0 Retention Control
0x210	SYSCFG_DMEM0ECCADDR	RH	DMEM0 ECC Address
0x214	SYSCFG_DMEM0ECCCTRL	RW	DMEM0 ECC Control
0x400	SYSCFG_RADIORAM-RETNCTRL	RW	RADIO RAM Retention Control Register
0x408	SYSCFG_RADIOECCCTRL	RW	RADIO RAM ECC Control Register
0x410	SYSCFG_SEQRAMECCADDR	RH	SEQRAM ECC Address
0x414	SYSCFG_FRCRAMECCADDR	RH	FRCRAM ECC Address
0x600	SYSCFG_ROOTDATA0	RW	Root Data Register 0
0x604	SYSCFG_ROOTDATA1	RW	Root Data Register 1
0x608	SYSCFG_ROOTLOCKSTATUS	RH	Lock Status
0x1000	SYSCFG_IF_SET	RWH INTFLAG	Interrupt Flag
0x1004	SYSCFG_IEN_SET	RW	Interrupt Enable
0x1010	SYSCFG_CHIPREVHW_SET	RWH	Chip Revision, Hard-wired
0x1014	SYSCFG_CHIPREV_SET	RW	Part Family and Revision Values
0x1200	SYSCFG_CTRL_SET	RW	Memory System Control
0x1208	SYSCFG_DMEM0RETNCTRL_SET	RW	DMEM0 Retention Control
0x1210	SYSCFG_DMEM0EC-CADDR_SET	RH	DMEM0 ECC Address
0x1214	SYSCFG_DMEM0ECCCTRL_SET	RW	DMEM0 ECC Control
0x1400	SYSCFG_RADIORAM-RETNCTRL_SET	RW	RADIO RAM Retention Control Register
0x1408	SYSCFG_RADIO-ECCCTRL_SET	RW	RADIO RAM ECC Control Register
0x1410	SYSCFG_SEQRAMEC-CADDR_SET	RH	SEQRAM ECC Address
0x1414	SYSCFG_FRCRAMEC-CADDR_SET	RH	FRCRAM ECC Address
0x1600	SYSCFG_ROOTDATA0_SET	RW	Root Data Register 0
0x1604	SYSCFG_ROOTDATA1_SET	RW	Root Data Register 1
0x1608	SYSCFG_ROOTLOCKSTA-TUS_SET	RH	Lock Status

Offset	Name	Type	Description
0x2000	SYSCFG_IF_CLR	RWH INTFLAG	Interrupt Flag
0x2004	SYSCFG_IEN_CLR	RW	Interrupt Enable
0x2010	SYSCFG_CHIPREVHW_CLR	RWH	Chip Revision, Hard-wired
0x2014	SYSCFG_CHIPREV_CLR	RW	Part Family and Revision Values
0x2200	SYSCFG_CTRL_CLR	RW	Memory System Control
0x2208	SYSCFG_DMEM0RETNCTRL_CLR	RW	DMEM0 Retention Control
0x2210	SYSCFG_DMEM0ECCADDR_CLR	RH	DMEM0 ECC Address
0x2214	SYSCFG_DMEM0ECCCTRL_CLR	RW	DMEM0 ECC Control
0x2400	SYSCFG_RADIORAMRETNCTRL_CLR	RW	RADIO RAM Retention Control Register
0x2408	SYSCFG_RADIOECCCTRL_CLR	RW	RADIO RAM ECC Control Register
0x2410	SYSCFG_SEQRAMECCADDR_CLR	RH	SEQRAM ECC Address
0x2414	SYSCFG_FRCRAMECCADDR_CLR	RH	FRCRAM ECC Address
0x2600	SYSCFG_ROOTDATA0_CLR	RW	Root Data Register 0
0x2604	SYSCFG_ROOTDATA1_CLR	RW	Root Data Register 1
0x2608	SYSCFG_ROOTLOCKSTATUS_CLR	RH	Lock Status
0x3000	SYSCFG_IF_TGL	RWH INTFLAG	Interrupt Flag
0x3004	SYSCFG_IEN_TGL	RW	Interrupt Enable
0x3010	SYSCFG_CHIPREVHW_TGL	RWH	Chip Revision, Hard-wired
0x3014	SYSCFG_CHIPREV_TGL	RW	Part Family and Revision Values
0x3200	SYSCFG_CTRL_TGL	RW	Memory System Control
0x3208	SYSCFG_DMEM0RETNCTRL_TGL	RW	DMEM0 Retention Control
0x3210	SYSCFG_DMEM0ECCADDR_TGL	RH	DMEM0 ECC Address
0x3214	SYSCFG_DMEM0ECCCTRL_TGL	RW	DMEM0 ECC Control
0x3400	SYSCFG_RADIORAMRETNCTRL_TGL	RW	RADIO RAM Retention Control Register
0x3408	SYSCFG_RADIOECCCTRL_TGL	RW	RADIO RAM ECC Control Register
0x3410	SYSCFG_SEQRAMECCADDR_TGL	RH	SEQRAM ECC Address
0x3414	SYSCFG_FRCRAMECCADDR_TGL	RH	FRCRAM ECC Address
0x3600	SYSCFG_ROOTDATA0_TGL	RW	Root Data Register 0

Offset	Name	Type	Description
0x3604	SYSCFG_ROOTDATA1_TGL	RW	Root Data Register 1
0x3608	SYSCFG_ROOTLOCKSTA-TUS_TGL	RH	Lock Status

6.6.6 SYSCFG Register Description

6.6.6.1 SYSCFG_IF - Interrupt Flag

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x0	0x0			0x0	0x0							0x0	0x0													0x0	0x0	0x0	0x0
Access			RW	RW			RW	RW							RW	RW													RW	RW	RW	RW
Name			FRCRAMERR2B	FRCRAMERR1B			SEQRAMERR2B	SEQRAMERR1B							RAMERR2B	RAMERR1B													SW3	SW2	SW1	SW0

Bit	Name	Reset	Access	Description
31:30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	FRCRAMERR2B	0x0	RW	**FRCRAM 2-Bit Error Interrupt Flag**
	FRCRAM 2-bit ECC Error Interrupt flag.			
28	FRCRAMERR1B	0x0	RW	**FRCRAM 1-Bit Error Interrupt Flag**
	FRCRAM 1-bit ECC Error Interrupt flag.			
27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	SEQRAMERR2B	0x0	RW	**SEQRAM 2-Bit Error Interrupt Flag**
	SEQRAM 2-bit ECC Error Interrupt flag.			
24	SEQRAMERR1B	0x0	RW	**SEQRAM 1-Bit Error Interrupt Flag**
	SEQRAM 1-bit ECC Error Interrupt flag.			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	RAMERR2B	0x0	RW	**RAM 2-Bit Error Interrupt Flag**
	RAM 2-bit ECC Error Interrupt flag.			
16	RAMERR1B	0x0	RW	**RAM 1-Bit Error Interrupt Flag**
	RAM 1-bit ECC Error Interrupt flag.			
15:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	SW3	0x0	RW	**Software Interrupt 3**
	Software interrupts			
2	SW2	0x0	RW	**Software Interrupt 2**
	Software interrupts			
1	SW1	0x0	RW	**Software Interrupt 1**
	Software interrupts			
0	SW0	0x0	RW	**Software Interrupt 0**

Bit	Name	Reset	Access	Description
	Software interrupts			

6.6.6.2 SYSCFG_IEN - Interrupt Enable

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x0	0x0			0x0	0x0							0x0	0x0													0x0	0x0	0x0	0x0
Access			RW	RW			RW	RW							RW	RW													RW	RW	RW	RW
Name			FRCRAMERR2B	FRCRAMERR1B			SEQRAMERR2B	SEQRAMERR1B							RAMERR2B	RAMERR1B													SW3	SW2	SW1	SW0

Bit	Name	Reset	Access	Description
31:30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	FRCRAMERR2B	0x0	RW	**FRCRAM 2-bit Error Interrupt Enable**
	Set to enable the FRCRAM2ERR2BIF Interrupt			
28	FRCRAMERR1B	0x0	RW	**FRCRAM 1-bit Error Interrupt Enable**
	Set to enable the FRCRAM2ERR1BIF Interrupt			
27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	SEQRAMERR2B	0x0	RW	**SEQRAM 2-bit Error Interrupt Enable**
	Set to enable the SEQRAM2ERR2BIF Interrupt			
24	SEQRAMERR1B	0x0	RW	**SEQRAM 1-bit Error Interrupt Enable**
	Set to enable the SEQRAM2ERR1BIF Interrupt			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	RAMERR2B	0x0	RW	**RAM 2-bit Error Interrupt Enable**
	Set to enable the RAMERR2BIF Interrupt			
16	RAMERR1B	0x0	RW	**RAM 1-bit Error Interrupt Enable**
	Set to enable the RAMERR1BIF Interrupt			
15:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	SW3	0x0	RW	**Software interrupt 3**
	Set to enable the Software Interrupts			
2	SW2	0x0	RW	**Software interrupt 2**
	Set to enable the Software Interrupts			
1	SW1	0x0	RW	**Software interrupt 1**
	Set to enable the Software Interrupts			
0	SW0	0x0	RW	**Software interrupt 0**
	Set to enable the Software Interrupts			

6.6.6.3 SYSCFG_CHIPREVHW - Chip Revision, Hard-wired

Offset	Bit Position																															
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0								0x30						0x1					
Access													RW								RW						RW					
Name													MINOR								FAMILY						MAJOR					

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:12	MINOR	0x0	RW	**Hardwired Chip Revision Minor value**
	Hardwired Chip Revision Minor signal value			
11:6	FAMILY	0x30	RW	**Hardwired Chip Family value**
	Hardwired Chip Family signal value			
5:0	MAJOR	0x1	RW	**Hardwired Chip Revision Major value**
	Hardwired Chip Revision Major signal value			

6.6.6.4 SYSCFG_CHIPREV - Part Family and Revision Values

Offset	Bit Position																															
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0								0x0						0x0					
Access													RW								RW						RW					
Name													MINOR								FAMILY						MAJOR					

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:12	MINOR	0x0	RW	**Chip Revision Minor value**
	Chip Revision Minor value			
11:6	FAMILY	0x0	RW	**Chip Family value**
	Chip Family value			
5:0	MAJOR	0x0	RW	**Chip Revision Major value**
	Chip Revision Major value			

6.6.6.5 SYSCFG_CTRL - Memory System Control

Offset	Bit Position																													
0x200	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x1				0x1
Access																											RW					RW
Name																											RAMECCERRFAULTEN					ADDRFAULTEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	RAMECCERRFAULTEN	0x1	RW	**Two bit ECC Error Bus Fault Response Enable**
	When this bit is set, busfaults are generated if a 2-bit ECC error occurs.			
4:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	ADDRFAULTEN	0x1	RW	**Invalid Address Bus Fault Response Enable**
	When this bit is set, busfaults are generated on accesses to unmapped parts of system and code address space			

6.6.6.6 SYSCFG_DMEM0RETNCTRL - DMEM0 Retention Control

Offset	Bit Position																													
0x208	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	
Access																															RW	
Name																															RAMRETNCTRL	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	RAMRETNCTRL	0x0	RW	**DMEM0 blockset retention control**

DMEM0 RAM blockset retention controls in EM23 with full access in EM01.

Value	Mode	Description
0	ALLON	None of the RAM blocks powered down
1	BLK0	Power down RAM block 0
2	BLK1	Power down RAM block 1

6.6.6.7 SYSCFG_DMEM0ECCADDR - DMEM0 ECC Address

Offset	Bit Position																													
0x210	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0																
Access																R																
Name																DMEM0ECCADDR																

Bit	Name	Reset	Access	Description
31:0	DMEM0ECCADDR	0x0	R	**DMEM0 RAM ECC Error Address**

Indicates the address in system RAM at which an ECC error has occurred.

6.6.6.8 SYSCFG_DMEM0ECCCTRL - DMEM0 ECC Control

Offset	Bit Position																													
0x214	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															RAMECCEWEN	RAMECCEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	RAMECCEWEN	0x0	RW	**RAM ECC Error Writeback Enable**
	When set, 1-bit ECC errors will be corrected and written back to RAM when encountered.			
0	RAMECCEN	0x0	RW	**RAM ECC Enable**
	When set, this bit enables the generation and checking of ECC for RAM.			

6.6.6.9 SYSCFG_RADIORAMRETNCTRL - RADIO RAM Retention Control Register

Offset	Bit Position																													
0x400	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0							0x0	
Access																								RW							RW	
Name																								FRCRAMRETNCTRL							SEQRAMRETNCTRL	

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	FRCRAMRETNCTRL	0x0	RW	**FRCRAM Retention Control**

FRC RAM power-down in EM23 with full access in EM01

Value	Mode	Description
0	ALLON	FRCRAM not powered down
1	ALLOFF	Power down FRCRAM

Bit	Name	Reset	Access	Description
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	SEQRAMRETNCTRL	0x0	RW	**SEQRAM Retention Control**

SEQUENCER RAM power-down in EM23 with full access in EM01

Value	Mode	Description
0	ALLON	SEQRAM not powered down
1	BLK0	Power down SEQRAM block 0
2	BLK1	Power down SEQRAM block 1
3	ALLOFF	Power down all SEQRAM blocks

6.6.6.10 SYSCFG_RADIOECCCTRL - RADIO RAM ECC Control Register

Offset	Bit Position																													
0x408	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0	0x0							0x0	0x0
Access																							RW	RW							RW	RW
Name																							FRCRAMECCEWEN	FRCRAMECCEN							SEQRAMECCEWEN	SEQRAMECCEN

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	FRCRAMECCEWEN	0x0	RW	**FRCRAM ECC Error Writeback Enable**
	FRC ECC Error Writeback Enable. When set, errors will be corrected when encountered.			
8	FRCRAMECCEN	0x0	RW	**FRCRAM ECC Enable**
	FRCRAM ECC Enable.			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	SEQRAMECCEWEN	0x0	RW	**SEQRAM ECC Error Writeback Enable**
	SEQRAM ECC Error Writeback Enable. When set, errors will be corrected when encountered.			
0	SEQRAMECCEN	0x0	RW	**SEQRAM ECC Enable**
	SEQRAM ECC Enable.			

6.6.6.11 SYSCFG_SEQRAMECCADDR - SEQRAM ECC Address

Offset	Bit Position																															
0x410	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	SEQRAMECCADDR																															

Bit	Name	Reset	Access	Description
31:0	SEQRAMECCADDR	0x0	R	**SEQRAM ECC Address**
	Indicates the address in SEQRAM at which ECC error occurred.			

6.6.6.12 SYSCFG_FRCRAMECCADDR - FRCRAM ECC Address

Offset	Bit Position																															
0x414	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	FRCRAMECCADDR																															

Bit	Name	Reset	Access	Description
31:0	FRCRAMECCADDR	0x0	R	**FRCRAM ECC Error Address**
	Indicates the address in FRCRAM at which an ECC error occurred.			

6.6.6.13 SYSCFG_ROOTDATA0 - Root Data Register 0

Offset	Bit Position																															
0x600	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x0	RW	**Data**
	Generic data space for user to pass to root, e.g., address of struct in mem			

6.6.6.14 SYSCFG_ROOTDATA1 - Root Data Register 1

Offset	Bit Position																															
0x604	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x0	RW	**Data**
	Generic data space for user to pass to root, e.g., address of struct in mem			

6.6.6.15 SYSCFG_ROOTLOCKSTATUS - Lock Status

Offset	Bit Position																														
0x608	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x1				0x1	0x1	0x1	0x1	0x1								0x1				0x0		0x1	0x1	0x1
Access								R				R	R	R	R	R								R				R		R	R	R
Name								RADIODBGLOCK				USERDBGAPLOCK	USERSPNIDLOCK	USERSPIDLOCK	USERNIDLOCK	USERDBGLOCK								ROOTDBGLOCK				ROOTMODELOCK		MFRLOCK	REGLOCK	BUSLOCK

Bit	Name	Reset	Access	Description
31:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	RADIODBGLOCK	0x1	R	**Radio Debug Lock**
	Locked when 1; unlocked when 0.			
23:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	USERDBGAPLOCK	0x1	R	**User Debug Access Port Lock**
	Locked when 1; unlocked when 0.			
19	USERSPNIDLOCK	0x1	R	**User Secure Non-invasive Debug Lock**
	Locked when 1; unlocked when 0.			
18	USERSPIDLOCK	0x1	R	**User Secure Invasive Debug Lock**
	Locked when 1; unlocked when 0.			
17	USERNIDLOCK	0x1	R	**User Non-invasive Debug Lock**
	Locked when 1; unlocked when 0.			
16	USERDBGLOCK	0x1	R	**User Invasive Debug Lock**
	Locked when 1; unlocked when 0.			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	ROOTDBGLOCK	0x1	R	**Root Debug Lock**
	Locked when 1; unlocked when 0.			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	ROOTMODELOCK	0x0	R	**Root Mode Lock**
	Locked when 1; unlocked when 0.			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	MFRLOCK	0x1	R	**Manufacture Lock**
	Locked when 1; unlocked when 0.			

Bit	Name	Reset	Access	Description
1	REGLOCK	0x1	R	**Register Lock**
	Locked when 1; unlocked when 0.			
0	BUSLOCK	0x1	R	**Bus Lock**
	Locked when 1; unlocked when 0.			

6.7 MSC Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	MSC_IPVERSION	R	IP version ID
0x008	MSC_READCTRL	RW	Read Control Register
0x00C	MSC_WRITECTRL	RW	Write Control Register
0x010	MSC_WRITECMD	W	Write Command Register
0x014	MSC_ADDRB	RW	Page Erase/Write Address Buffer
0x018	MSC_WDATA	RW	Write Data Register
0x01C	MSC_STATUS	RH	Status Register
0x020	MSC_IF	RWH	Interrupt Flag Register
0x024	MSC_IEN	RW	Interrupt Enable Register
0x034	MSC_USERDATASIZE	R	User Data Region Size Register
0x038	MSC_CMD	W	Command Register
0x03C	MSC_LOCK	W	Configuration Lock Register
0x040	MSC_MISCLOCKWORD	RW	Mass erase and User data page lock word
0x050	MSC_PWRCTRL	RW	Power control register
0x120	MSC_PAGELOCK0	RW	Main space page 0-31 lock word
0x124	MSC_PAGELOCK1	RW	Main space page 32-63 lock word
0x1000	MSC_IPVERSION_SET	R	IP version ID
0x1008	MSC_READCTRL_SET	RW	Read Control Register
0x100C	MSC_WRITECTRL_SET	RW	Write Control Register
0x1010	MSC_WRITECMD_SET	W	Write Command Register
0x1014	MSC_ADDRB_SET	RW	Page Erase/Write Address Buffer
0x1018	MSC_WDATA_SET	RW	Write Data Register
0x101C	MSC_STATUS_SET	RH	Status Register
0x1020	MSC_IF_SET	RWH	Interrupt Flag Register
0x1024	MSC_IEN_SET	RW	Interrupt Enable Register
0x1034	MSC_USERDATASIZE_SET	R	User Data Region Size Register
0x1038	MSC_CMD_SET	W	Command Register
0x103C	MSC_LOCK_SET	W	Configuration Lock Register
0x1040	MSC_MISCLOCKWORD_SET	RW	Mass erase and User data page lock word
0x1050	MSC_PWRCTRL_SET	RW	Power control register
0x1120	MSC_PAGELOCK0_SET	RW	Main space page 0-31 lock word
0x1124	MSC_PAGELOCK1_SET	RW	Main space page 32-63 lock word
0x2000	MSC_IPVERSION_CLR	R	IP version ID
0x2008	MSC_READCTRL_CLR	RW	Read Control Register
0x200C	MSC_WRITECTRL_CLR	RW	Write Control Register

Offset	Name	Type	Description
0x2010	MSC_WRITECMD_CLR	W	Write Command Register
0x2014	MSC_ADDRB_CLR	RW	Page Erase/Write Address Buffer
0x2018	MSC_WDATA_CLR	RW	Write Data Register
0x201C	MSC_STATUS_CLR	RH	Status Register
0x2020	MSC_IF_CLR	RWH	Interrupt Flag Register
0x2024	MSC_IEN_CLR	RW	Interrupt Enable Register
0x2034	MSC_USERDATASIZE_CLR	R	User Data Region Size Register
0x2038	MSC_CMD_CLR	W	Command Register
0x203C	MSC_LOCK_CLR	W	Configuration Lock Register
0x2040	MSC_MISCLOCKWORD_CLR	RW	Mass erase and User data page lock word
0x2050	MSC_PWRCTRL_CLR	RW	Power control register
0x2120	MSC_PAGELOCK0_CLR	RW	Main space page 0-31 lock word
0x2124	MSC_PAGELOCK1_CLR	RW	Main space page 32-63 lock word
0x3000	MSC_IPVERSION_TGL	R	IP version ID
0x3008	MSC_READCTRL_TGL	RW	Read Control Register
0x300C	MSC_WRITECTRL_TGL	RW	Write Control Register
0x3010	MSC_WRITECMD_TGL	W	Write Command Register
0x3014	MSC_ADDRB_TGL	RW	Page Erase/Write Address Buffer
0x3018	MSC_WDATA_TGL	RW	Write Data Register
0x301C	MSC_STATUS_TGL	RH	Status Register
0x3020	MSC_IF_TGL	RWH	Interrupt Flag Register
0x3024	MSC_IEN_TGL	RW	Interrupt Enable Register
0x3034	MSC_USERDATASIZE_TGL	R	User Data Region Size Register
0x3038	MSC_CMD_TGL	W	Command Register
0x303C	MSC_LOCK_TGL	W	Configuration Lock Register
0x3040	MSC_MISCLOCKWORD_TGL	RW	Mass erase and User data page lock word
0x3050	MSC_PWRCTRL_TGL	RW	Power control register
0x3120	MSC_PAGELOCK0_TGL	RW	Main space page 0-31 lock word
0x3124	MSC_PAGELOCK1_TGL	RW	Main space page 32-63 lock word

6.8 MSC Register Description

6.8.1 MSC_IPVERSION - IP version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x1	R	**IP Version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

6.8.2 MSC_READCTRL - Read Control Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x2									0x0											
Access											RW									RW											
Name											MODE									DOUTBUFEN											

Bit	Name	Reset	Access	Description
31:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	MODE	0x2	RW	**Read Mode**

When changing to a higher frequency, this register must be set to a large number of wait states before the core clock is switched to the higher frequency. When changing to a lower frequency, this register should be set to a lower number of wait states after the frequency transition has been completed The maximum frequency for each wait state setting is listed in the datasheet.

Value	Mode	Description
0	WS0	Zero wait-states inserted in fetch or read transfers
1	WS1	One wait-state inserted for each fetch or read transfer
2	WS2	Two wait-states inserted for eatch fetch or read transfer
3	WS3	Three wait-states inserted for eatch fetch or read transfer

19:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	DOUTBUFEN	0x0	RW	**Flash dout pipeline buffer enable**

Flash dout buffer prefetch enable. Once disabled, every read will be a new flash read operation even the new read from the same flash entry as previous read (prefetch hit).

11:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

6.8.3 MSC_WRITECTRL - Write Control Register

Offset	Bit Position																															
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		0x0	0x0
Access																													RW		RW	RW
Name																													LPWRITE		IRQERASEABORT	WREN

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	LPWRITE	0x0	RW	**Low-Power Erase** When set, user write times might double while reducing current consumption
2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	IRQERASEABORT	0x0	RW	**Abort Page Erase on Interrupt** When this bit is set to 1, any Cortex-M33 interrupt aborts any current page erase operation. Executing that interrupt vector from flash will halt the CPU.
0	WREN	0x0	RW	**Enable Write/Erase Controller** When this bit is set, the MSC User write and erase functionality is enabled

6.8.4 MSC_WRITECMD - Write Command Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																				0x0				0x0			0x0			0x0	0x0	
Access																				W				W			W			W	W	
Name																				CLEARWDATA				ERASEMAIN0			ERASEABORT			WRITEEND	ERASEPAGE	

Bit	Name	Reset	Access	Description
31:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	CLEARWDATA	0x0	W	**Clear WDATA state**
	Will set WDATAREADY and DMA request. Should only be used when no write is active.			
11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	ERASEMAIN0	0x0	W	**Mass erase region 0**
	Initiate mass erase of flash memory. Before use, MSC_MASSLOCK must be unlocked. To completely prevent access from software, clear bit 0 in the mass erase lock-word (MLW)			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	ERASEABORT	0x0	W	**Abort erase sequence**
	Writing to this bit will abort an ongoing erase sequence.			
4:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	WRITEEND	0x0	W	**End Write Mode**
	Write 1 to abort a write command.			
1	ERASEPAGE	0x0	W	**Erase Page**
	Erase any user defined page selected by the MSC_ADDRB register. The WREN bit in the MSC_WRITECTRL register must be set in order to use this command.			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

6.8.5 MSC_ADDRB - Page Erase/Write Address Buffer

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	ADDRB																															

Bit	Name	Reset	Access	Description
31:0	ADDRB	0x0	RW	**Page Erase or Write Address Buffer**
	This register holds the system address for the erase or write operation. Address should be word aligned address.The MSB bit is not ignored for ADDRB			

6.8.6 MSC_WDATA - Write Data Register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	DATAW																															

Bit	Name	Reset	Access	Description
31:0	DATAW	0x0	RW	**Write Data**
	The data to be written to the address in MSC_ADDR. This register must be written when the WDATAREADY bit of MSC_STATUS is set.This register does not support write mask.			

6.8.7 MSC_STATUS - Status Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0				0x1			0x0								0x0										0x0	0x0	0x0	0x1	0x0	0x0	0x0
Access	R				R			R								R										R	R	R	R	R	R	R
Name	PWRUPCKBDFAILCOUNT				WREADY			PWRON								REGLOCK										TIMEOUT	PENDING	ERASEABORTED	WDATAREADY	INVADDR	LOCKED	BUSY

Bit	Name	Reset	Access	Description
31:28	PWRUPCKBDFAIL-COUNT	0x0	R	**Flash power up checkerboard pattern chec**
	This field tells how many times checkboard pattern check fail occured after a reset sequence.			
27	WREADY	0x1	R	**Flash Write Ready**
	When this bit is set, flash has completed the power up sequence and is ready for write/erase commands.			
26:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	PWRON	0x0	R	**Flash Power On Status**
	When this bit is set, flash is powered on. If zero, flash is powered off and reads from flash return indeterminate data.			
23:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	REGLOCK	0x0	R	**Register Lock Status**
	Indicates the current status of register lock			

Value	Mode	Description
0	UNLOCKED	Register lock is unlocked
1	LOCKED	Register lock is locked.

15:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	TIMEOUT	0x0	R	**Write Command Timeout**
	When this bit is set, it indicates that the last write command has completed due to a write buffer timeout. This bit is cleared automatically when a new write command is initiated.			
5	PENDING	0x0	R	**Write Command In Queue**
	When this bit is set, a flash operation has been requested but not yet started. New commands are ignored when PENDING is set.			
4	ERASEABORTED	0x0	R	**Erase Operation Aborted**
	When MSC_WRITECTRL_IRQERASEABORT = 1, this bit is set because an interrupt has aborted the erase operation in progress.			

Bit	Name	Reset	Access	Description
3	WDATAREADY	0x1	R	**WDATA Write Ready**
	When this bit is set, the content of MSC_WDATA is read by MSC Flash Write Controller and the register may be updated with the next 32-bit word to be written to flash. This bit is cleared when writing to MSC_WDATA.			
2	INVADDR	0x0	R	**Invalid Write Address or Erase Page**
	When this bit is set, software has attempted to load an invalid (unmapped) address into the MSC_ADDRB register.			
1	LOCKED	0x0	R	**Access Locked**
	When set, the last erase or write was aborted due to erase/write access constraints.			
0	BUSY	0x0	R	**Erase/Write Busy**
	When set, an erase or write operation is in progress and new commands are ignored.			

6.8.8 MSC_IF - Interrupt Flag Register

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0	0x0						0x0	0x0	0x0
Access																							RW	RW						RW	RW	RW
Name																							PWROFF	PWRUPF						WDATAOV	WRITE	ERASE

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	PWROFF	0x0	RW	**Flash Power Off Sequence Complete Flag**
	Set after MSC_CMD.PWROFF received, flash powered off complete			
8	PWRUPF	0x0	RW	**Flash Power Up Sequence Complete Flag**
	Set after MSC_CMD.PWRUP received, flash powered up complete and ready for read/write			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	WDATAOV	0x0	RW	**Host write buffer overflow**
	If set, flash controller write buffer overflow detected			
1	WRITE	0x0	RW	**Host Write Done Interrupt Read Flag**
	Set when a write is done			
0	ERASE	0x0	RW	**Host Erase Done Interrupt Read Flag**
	Set when erase is done			

6.8.9 MSC_IEN - Interrupt Enable Register

Offset	Bit Position																														
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0	0x0						0x0	0x0	0x0
Access																							RW	RW						RW	RW	RW
Name																							PWROFF	PWRUPF						WDATAOV	WRITE	ERASE

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	PWROFF interrupt enable	0x0	RW	**Flash Power Off Seq done irq enable**
8	PWRUPF interrupt enable	0x0	RW	**Flash Power Up Seq done irq enable**
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	WDATAOV interrupt enable	0x0	RW	**write data buffer overflow irq enable**
1	WRITE interrupt enable	0x0	RW	**Write Done Interrupt enable**
0	ERASE interrupt enable	0x0	RW	**Erase Done Interrupt enable**

6.8.10 MSC_USERDATASIZE - User Data Region Size Register

Offset	Bit Position																															
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x4				
Access																											R					
Name																											USERDATASIZE					

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	USERDATASIZE	0x4	R	**User Data Size**
	This field determines user data region size. SIZE = 256B * USERDATASIZE.			

6.8.11 MSC_CMD - Command Register

Offset	Bit Position																															
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				0x0
Access																												W				W
Name																												PWROFF				PWRUP

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	PWROFF	0x0	W	**Flash power off/sleep command**
	Write to this bit to power down the Flash. User code should execute from RAM afterwards. Read from flash after flash being powered down will cause undetermined behavior. To power up, either set CMD.PWRUP bit or try read from flash.			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	PWRUP	0x0	W	**Flash Power Up Command**
	Write to this bit to power up the Flash. IRQ PWRUPF will be fired when power up sequence completed.			

6.8.12 MSC_LOCK - Configuration Lock Register

Offset	Bit Position																															
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	W															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x0	W	**Configuration Lock**

Write any other value than the unlock code to lock access to MSC_CTRL, MSC_READCTRL, MSC_WRITECTRL. Write the unlock code to enable access. When reading the register, bit 0 is set when the lock is enabled.

Value	Mode	Description
0	LOCK	Key to lock the register lock
7025	UNLOCK	Key to unlock the register lock.

6.8.13 MSC_MISCLOCKWORD - Mass erase and User data page lock word

Offset	Bit Position																															
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x1				0x1
Access																												RW				RW
Name																												UDLOCKBIT				MELOCKBIT

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	UDLOCKBIT	0x1	RW	**User Data Lock**

Zero means host can write to the user data area. Host is only allowed to write one. Root and debug can clear this bit.

3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	MELOCKBIT	0x1	RW	**Mass Erase Lock**

Zero means host can mass erase the main space. Host is only allowed to write one. Root and debug can clear this bit.

6.8.14 MSC_PWRCTRL - Power control register

Offset	Bit Position																													
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x10																0x0			0x1	0x0
Access												RW																RW			RW	RW
Name												PWROFFDLY																PWROFFENTRYAGAIN			PWROFFONEM1PENTRY	PWROFFONEM1ENTRY

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:16	PWROFFDLY	0x10	RW	**Power down delay**
	Defines delay cycles before flash enters sleep mode. Works together with PWROFFENTRYAGAIN bit. The power off delay is 64 * PWROFFDLY bus clock cycles.			
15:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	PWROFFENTRYAGAIN	0x0	RW	**POWER down flash again in EM1/EM1p**
	If enabled, flash will enter sleep mode again when POWEROFFONEM1ENTRY/POWEROFFONEM1PENTRY is set and no flash activities occur for the time determined by PWROFFSTDLY.			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	PWROFFONEM1PENTRY	0x1	RW	**Power down Flash macro when enter EM1P**
	If enabled, flash will be in sleep mode when entering EM1P (radio-only sleep).			
0	PWROFFONEM1ENTRY	0x0	RW	**Power down Flash macro when enter EM1**
	If enabled, flash will be in sleep mode when entering EM1.			

6.8.15 MSC_PAGELOCK0 - Main space page 0-31 lock word

Offset	Bit Position																													
0x120	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	LOCKBIT																															

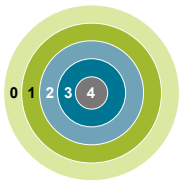
Bit	Name	Reset	Access	Description
31:0	LOCKBIT	0x0	RW	**page lock bit**
	Zero means the corresponding page is allowed to write/erase. change to one will prevent corresponding page from write/erase. bit[0] for main space page 0, and bit[1] for page 1... bit[31] for page 31. Reset to zero. Host is only allowed to write one. Root and Debug are allowed to clear this register			

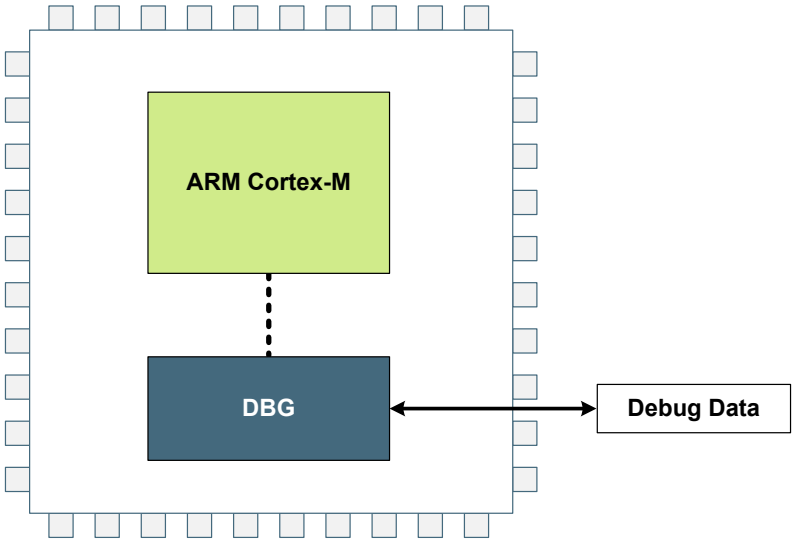
6.8.16 MSC_PAGELOCK1 - Main space page 32-63 lock word

Offset	Bit Position																													
0x124	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	LOCKBIT																															

Bit	Name	Reset	Access	Description
31:0	LOCKBIT	0x0	RW	**page lock bit**
	Zero means the corresponding page is allowed to write/erase. change to one will prevent corresponding page from write/erase. bit[0] for main space page 32, and bit[1] for page 33... bit[31] for page 63. Reset to zero. Host is only allowed to write one. Root and Debug are allowed to clear this register			

7. DBG - Debug Interface





Quick Facts

What?

The Debug Interface is used to program and debug EFR32xG22 devices.

Why?

The Debug Interface makes it easy to re-program and update the system in the field, and allows debugging with minimal I/O pin usage.

How?

The Cortex-M33 supports advanced debugging features. EFR32xG22 devices can use a minimum of two port pins for debugging or programming. The internal and external state of the system can be examined with debug extensions supporting instruction or data access break and watch points.

7.1 Introduction

The EFR32xG22 devices include hardware debug support through a 2-pin serial-wire debug (SWD) interface or a 4-pin Joint Test Action Group (JTAG) interface, as well as an Embedded Trace Module (ETM) for data/instruction tracing. In addition, there is also a Serial Wire Viewer pin which can be used to output profiling information, data trace and software-generated messages.

For more technical information about the debug interface the reader is referred to:

- ARM Cortex-M33 Technical Reference Manual
- ARM CoreSight Components Technical Reference Manual
- ARM Debug Interface v5 Architecture Specification
- IEEE Standard for Test Access Port and Boundary-Scan Architecture, IEEE 1149.1-2013

7.2 Features

- Debug Access Port Serial Wire JTAG (DAPSWJ)
  - Implements the ADIv5 debug interface
- ARM Trustzone
  - Enables secure debugging
- Breakpoint unit (BPU)
  - Implement up to 8 hardware breakpoints
- Data Watch point and Trace (DWT) unit
  - Implement up to 4 watch points, trigger resources and system profiling
- Instrumentation Trace Macrocell (ITM)
  - Application-driven trace source that supports printf style debugging
- Embedded Trace Macrocell v3.5 (ETM)
  - Real time instruction and data trace information of the processor
- Cross Trigger Interface (CTI)
  - Issues synchronous triggers based on system events
  - Can be used to generate IRQs or route to PRS signalling

7.3 Functional Description

There are debug and trace pins available on the device. Operation of these pins is described in the following sections.

7.3.1 Debug Pins

The following pins are the debug connections for the device:

- Serial Wire Clock Input and Test Clock Input (SWCLKTCK) (SWCLK) : This pin is enabled after power-up and has a built-in pull-down.
- Serial Wire Data Input/Output and Test Mode Select Input (SWDIOTMS) (SWDIO) : This pin is enabled after power-up and has a built-in pull-up.
- Test Data Output (TDO): This pin is assigned to JTAG functionality after power-up. However, it remains in high-Z state until the first valid JTAG command is received.
- Test Data Input (TDI): This pin is assigned to JTAG functionality after power-up. However, it remains in high-Z state until the first valid JTAG command is received. Once enabled, the pin has a built-in pull-up.
- Serial Wire Viewer (SWV): This pin is disabled after reset.

The debug pins have integrated pull devices that are enabled by default after a reset. Leaving them enabled may increase current consumption if the pins are connected to power or ground. The debug pins have enable bits in the GPIO_DBGROUTEPEN register; refer to the GPIO chapter for more details. Upon disabling the debug pins, debug contact with the device is lost once the DAPSWJ power request bits are deasserted. By default after a power cycle, the DAPSWJ is in JTAG mode. If during a debugging session the device is switched to SWD mode, a power cycle is needed to return to JTAG mode.

7.3.2 Embedded Trace Macrocell V3.5 (ETM)

ETM makes it possible to non-intrusively trace both instruction and data from the processor in real time. Trace can be controlled through a set of triggering and filtering resources. The resources include 4 address comparators, 2 data value comparators, 2 counters, a context ID comparator and a sequencer. Before enabling the ETM, the CMU_TRACECLKCTRL register must be configured to select the desired trace clock source. (See the CMU chapter for details.)

The trace can be exported through a set of trace pins, which include:

- Trace Clock (TCLK): Functions as a sample clock for the trace. This pin is disabled after reset.
- Trace Data 0 (TD0): The trace data pin provides the compressed trace stream. This pin is disabled after reset.

For information on how to configure the ETM, see the ARM Embedded Trace Macrocell Architecture Specification. The Trace Clock and Trace Data pins are enabled through a GPIO register. For more information on how to enable the ETM pins, refer to the GPIO chapter.

7.3.3 Debug and EM2/EM3

Debug connectivity in EM2 and EM3 is unavailable by default, to reduce current consumption. Debugging through EM2 and EM3 can be enabled by setting the EM2DBGEN bit in the EMU_CTRL register. Setting EM2DBGEN ensures that power domain associated with the debug circuitry will remain active, but will result in a small amount of additional current in EM2 and EM3.

Leaving the debugger connected when issuing a WFI or WFE to enter EM2 or EM3 will make the system enter a special EM2 mode. This mode differs from regular EM2 and EM3 in that the high frequency clocks are still enabled, and certain core functionality is still powered in order to maintain debug functionality. Because of this, the current consumption in this mode is closer to EM1, and it is, therefore, important to deassert the power requests in the DAPSWJ and disconnect the debugger before undertaking current consumption measurements.

7.4 DBG Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x1000	DBG_DCIWDATA	RW	Write Data
0x1004	DBG_DCIRDATA	R	Read Data
0x1008	DBG_DCISTATUS	R	Status
0x10FC	DBG_DCIID	R	Identification
0x1110	DBG_SYSCOM0	R	Communication Status
0x1114	DBG_SYSCOM1	R	Communication Status
0x1120	DBG_SYSPWR0	R	Power Status
0x1130	DBG_SYSCLK0	R	Clocking Status
0x11FC	DBG_SYSID	R	Identification

7.5 DBG Register Description

7.5.1 DBG_DCIWDATA - Write Data

Offset	Bit Position																															
0x1000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	WDATA																															

Bit	Name	Reset	Access	Description
31:0	WDATA	0x0	RW	**Challenge Write Data**
	Data Sent to the Challenge Interface			

7.5.2 DBG_DCIRDATA - Read Data

Offset	Bit Position																															
0x1004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	RDATA																															

Bit	Name	Reset	Access	Description
31:0	RDATA	0x0	R	**Challenge Read Data**
	Data Response from the Challenge Interface			

7.5.3 DBG_DCISTATUS - Status

Offset	Bit Position																															
0x1008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0								0x0
Access																								R								R
Name																								RDATAVALID								WPENDING

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	RDATAVALID	0x0	R	**Read Data Valid** Response from the challenge interface is valid. Cleared on a read of DCIRDATA.
7:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	WPENDING	0x0	R	**Write Pending** Write Request to the Challenge interface is pending. Additional writes to DCIWDATA are discarded when this bit is asserted.

7.5.4 DBG_DCIID - Identification

Offset	Bit Position																															
0x10FC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0xDC11D																
Access																R																
Name																ID																

Bit	Name	Reset	Access	Description
31:0	ID	0xDC11D	R	**Identification** Debug Challenge Interface ID

7.5.5 DBG_SYSCOM0 - Communication Status

Offset	Bit Position																													
0x1110	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0													0x0	0x0	0x0	0x0								0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access	R													R	R	R	R								R	R	R	R	R	R	R	R
Name	PRS													RADIOSYNTHEN	RADIOPAEN	RADIOLNAEN	LDMACHACT								I2C1BSY	I2C0BSY	USART2TXBSY	USART2RXBSY	USART1TXBSY	USART1RXBSY	USART0TXBSY	USART0RXBSY

Bit	Name	Reset	Access	Description
31:20	PRS	0x0	R	**Peripheral Reflex Signals**
	Peripheral Reflex Signal Channel Value			
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	RADIOSYNTHEN	0x0	R	**RADIO Synthesizer Enabled**
	RF Synthesiszer Enabled for Radio TX and RX			
17	RADIOPAEN	0x0	R	**RADIO Power Amplifer Enabled**
	RF Power Amplifer Enabled for Radio TX			
16	RADIOLNAEN	0x0	R	**RADIO Low Noise Amplifer Enabled**
	RF Low Noise Amplifer Enabled for Radio RX			
15:8	LDMACHACT	0x0	R	**LDMA Channel Active**
	Active LDMA Channels			
7	I2C1BSY	0x0	R	**I2C1 Busy**
	I2C1 Transaction in progress			
6	I2C0BSY	0x0	R	**I2C0 Busy**
	I2C0 Transaction in progress			
5	USART2TXBSY	0x0	R	**USART2 Transmit Busy**
	USART2 Transmit Active			
4	USART2RXBSY	0x0	R	**USART2 Receive Busy**
	USART2 Receive Active			
3	USART1TXBSY	0x0	R	**USART1 Transmit Busy**
	USART1 Transmit Active			
2	USART1RXBSY	0x0	R	**USART1 Receive Busy**
	USART1 Receive Active			
1	USART0TXBSY	0x0	R	**USART0 Transmit Busy**
	USART0 Transmit Active			
0	USART0RXBSY	0x0	R	**USART0 Receive Busy**

Bit	Name	Reset	Access	Description
	USART0 Receive Active			

7.5.6 DBG_SYSCOM1 - Communication Status

Offset	Bit Position																															
0x1114	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x0													
Access																			R													
Name																			GPIO													

Bit	Name	Reset	Access	Description
31:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:0	GPIO	0x0	R	**General Purpose Input**
	General Purpose Input Value			

7.5.7 DBG_SYSPWR0 - Power Status

Offset	Bit Position																													
0x1120	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																							R	R	R	R	R	R	R	R	R	R
Name																							RTNDRAM1	RTNDRAM0	VS2	VS1	VS0	EM3	EM2	EM1P	EM1	EM0

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	RTNDRAM1	0x0	R	**Retained RAM0**
	Retained RAM0 Instance			
8	RTNDRAM0	0x0	R	**Retained RAM0**
	Retained RAM0 Instance			
7	VS2	0x0	R	**Voltage State 2**
	System is in Voltage State 2			
6	VS1	0x0	R	**Voltage State 1**
	System is in Voltage State 1			
5	VS0	0x0	R	**Voltage State 0**
	System is in Voltage State 0			
4	EM3	0x0	R	**Energy Mode 3**
	System is in Energy Mode 3			
3	EM2	0x0	R	**Energy Mode 2**
	System is in Energy Mode 2			
2	EM1P	0x0	R	**Energy Mode 1 Peripheral**
	System is in Energy Mode 1 Peripheral			
1	EM1	0x0	R	**Energy Mode 1**
	System is in Energy Mode 1			
0	EM0	0x0	R	**Energy Mode 0**
	System is in Energy Mode 0			

7.5.8 DBG_SYSCLK0 - Clocking Status

Offset	Bit Position																															
0x1130	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0	0x0	0x0		0x0	0x0	0x0	0x0		0x0	0x0	0x0		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access				R	R	R		R	R	R	R		R	R	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name				RNGULFRCO	RNGLFRCO	RNGLFXO		RNGHFRCODPLL	RNGSYXO	RNGFSRCO	RNGI2C0		RNGWDOG0	RNGPRORTC	RNGLETIMER0		RNGIADC	RNGDPLL	RNGLDMA	RNGGPCRC	RNGI2C1	RNGUSART2	RNGUSART1	RNGUSART0	RNGTIMER4	RNGTIMER3	RNGTIMER2	RNGTIMER1	RNGTIMER0	RNGRADIO	RNGRTCC	RNGBURTC

Bit	Name	Reset	Access	Description
31:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28	RNGULFRCO	0x0	R	**Running ULFRCO Clock**
	ULFRCO Clock is running			
27	RNGLFRCO	0x0	R	**Running LFRCO Clock**
	LFRCO Clock is running			
26	RNGLFXO	0x0	R	**Running LFXO Clock**
	LFXO Clock is running			
25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	RNGHFRCODPLL	0x0	R	**Running HFRCODPLL Clock**
	HFRCODPLL Clock is running			
23	RNGSYXO	0x0	R	**Running SYXO Clock**
	SYXO Clock is running			
22	RNGFSRCO	0x0	R	**Running FSRCO Clock**
	FSRCO Clock is running			
21	RNGI2C0	0x0	R	**Running I2C0 Clock**
	I2C0 Clock is running			
20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	RNGWDOG0	0x0	R	**Running WDOG0 Clock**
	WDOG0 Clock is running			
18	RNGPRORTC	0x0	R	**Running PRORTC Clock**
	PRORTC Clock is running			
17	RNGLETIMER0	0x0	R	**Running LETIMER0 Clock**
	LETIMER0 Clock is running			
16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

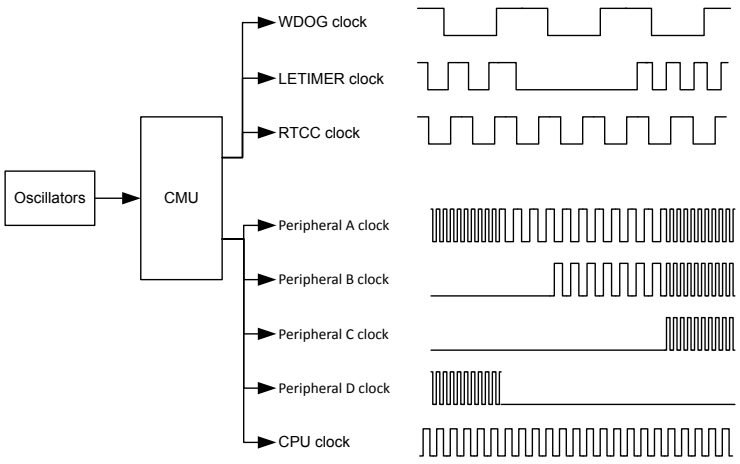
Bit	Name	Reset	Access	Description
15	RNGIADC	0x0	R	**Running IADC Clock**
	IADC Clock is running			
14	RNGDPLL	0x0	R	**Running DPLL Clock**
	DPLL Clock is running			
13	RNGLDMA	0x0	R	**Running LDMA Clock**
	LDMA Clock is running			
12	RNGGPCRC	0x0	R	**Running GPCRC Clock**
	GPCRC Clock is running			
11	RNGI2C1	0x0	R	**Running I2C1 Clock**
	I2C1 Clock is running			
10	RNGUSART2	0x0	R	**Running USART2 Clock**
	USART2 Clock is running			
9	RNGUSART1	0x0	R	**Running USART1 Clock**
	USART1 Clock is running			
8	RNGUSART0	0x0	R	**Running USART0 Clock**
	USART0 Clock is running			
7	RNGTIMER4	0x0	R	**Running TIMER4 Clock**
	TIMER4 Clock is running			
6	RNGTIMER3	0x0	R	**Running TIMER3 Clock**
	TIMER3 Clock is running			
5	RNGTIMER2	0x0	R	**Running TIMER2 Clock**
	TIMER2 Clock is running			
4	RNGTIMER1	0x0	R	**Running TIMER1 Clock**
	TIMER1 Clock is running			
3	RNGTIMER0	0x0	R	**Running TIMER0 Clock**
	TIMER0 Clock is running			
2	RNGRADIO	0x0	R	**Running RADIO Clock**
	RADIO Clock is running			
1	RNGRTCC	0x0	R	**Running RTCC Clock**
	RTCC Clock is running			
0	RNGBURTC	0x0	R	**Running BURTC Clock**
	BURTC Clock is running			

7.5.9 DBG_SYSID - Identification

Offset	Bit Position																													
0x11FC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x5451D																															
Access	R																															
Name	ID																															

Bit	Name	Reset	Access	Description
31:0	ID	0x5451D	R	**Identification**
	System Block ID			

8. CMU - Clock Management Unit



Quick Facts

What?

The CMU controls clock switching and distribution. EFR32xG22 supports 7 different oscillators with minimized power consumption and short start-up time. The CMU has HW support for calibration of RC oscillators.

Why?

Oscillators and clocks contribute significantly to the power consumption of the MCU. With the low power oscillators combined with the flexible clock control scheme, it is possible to minimize the energy consumption in any given application.

How?

The CMU switches different clock sources for various peripherals and sets the prescaler for the bus clocks. The short oscillator start-up times makes duty-cycling between active mode and the different low energy modes (EM2 DeepSleep, EM3 Stop, and EM4) very efficient. The calibration feature ensures high accuracy RC oscillators. Interrupt are available to avoid CPU polling of flags.

8.1 Introduction

The Clock Management Unit (CMU) is responsible for switching among various oscillator sources and provide clocks to the peripheral modules. Oscillators are automatically turned on and off based on demand from the peripherals to minimize power consumption.

8.2 Features

- Multiple clock sources available:
  - 38 MHz - 40 MHz High Frequency Crystal Oscillator (HFXO)
  - 1 MHz - 80 MHz High Frequency RC Oscillator (HFRCODPLL)
  - 20 MHz Fast Startup RC Oscillator (FSRCO)
  - 1 MHz - 38 MHz External Clock from Input Pins (CLKIN0)
  - 32.768 kHz Low Frequency Crystal Oscillator (LFXO)
  - 32.768 kHz Low Frequency RC Oscillator (LFRCO) with Precision Mode
  - 1000 Hz Ultra Low Frequency RC Oscillator (ULFRCO)
- On-demand oscillator request.
- Low power oscillators.
- Fast start-up times.
- Cascaded prescalers for AHB Clocks (HCLK) and APB Clocks (PCLK).
- Clock gating on an individual basis to all peripherals based on module enable.
- Reset on an individual basis for Timer and IADC based on module enable.
- Selectable clocks can be output on external pins and/or PRS.
- FSRCO, which is asynchronous to the system clock, can be selected for IADC operation in EM2.
- Hardware support for calibration of RC oscillators.

8.3 Functional Description

The CMU is comprised of several programmable clock trees, which connect oscillator resources to peripherals and buses. This section describes clock sources and peripherals available to the largest devices in the EFR32xG22 family. Please refer to the Configuration Summary in the Device Datasheet to see which core and peripheral modules, and therefore clock connections, are present in a specific device. Bus clock selection, including peripherals clocked directly from bus clocks, is shown in Figure 8.1 Bus Clocks on page 135. Clock selection for peripherals with multiple high-frequency clock sources is shown in Figure 8.2 High Frequency Peripheral Clocks on page 136. Clock selection for peripherals with multiple low-frequency clock sources is shown in Figure 8.3 Low Frequency Peripheral Clocks on page 137.

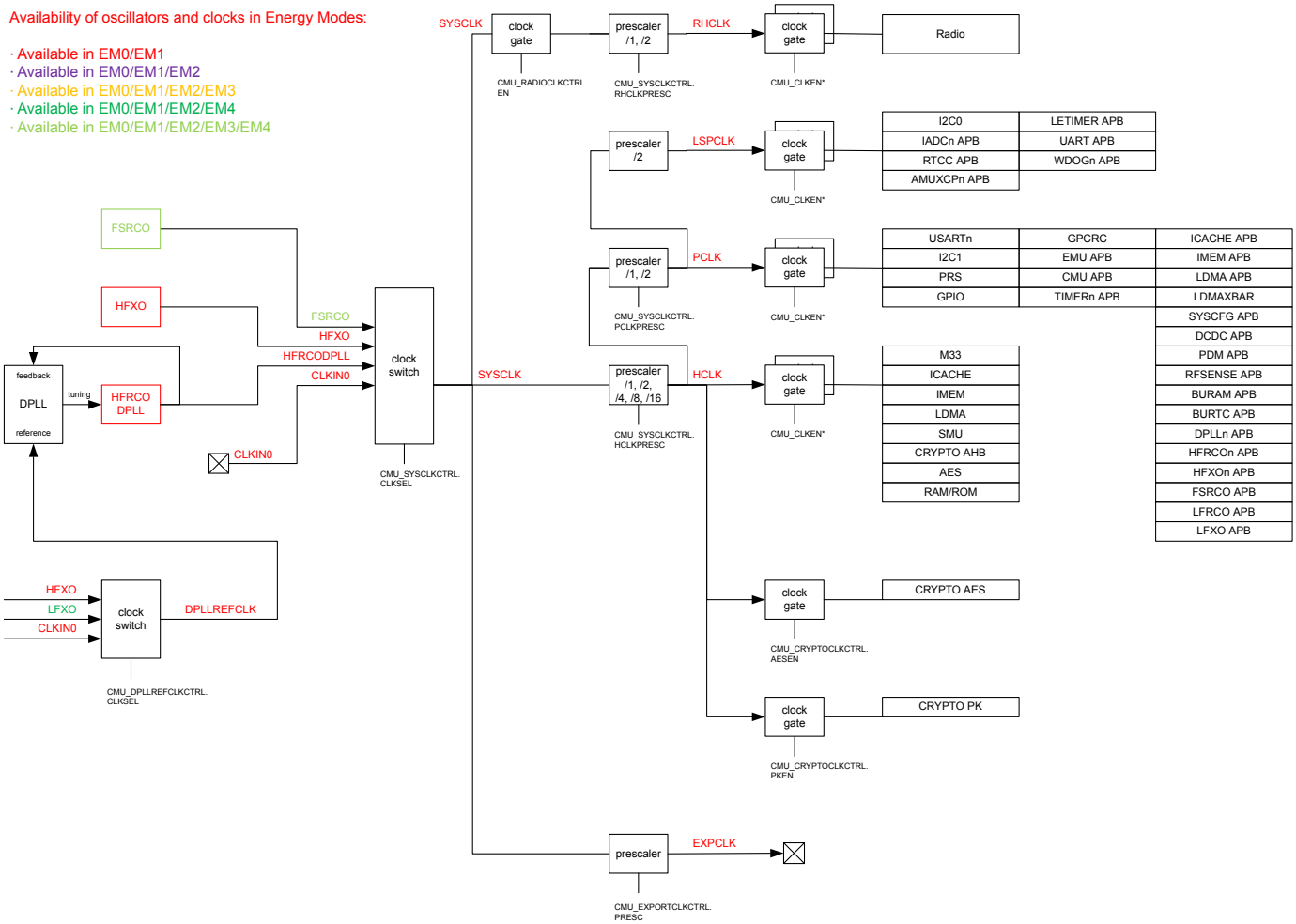


Figure 8.1. Bus Clocks

Availability of oscillators and clocks in Energy Modes:

- Available in EM0/EM1
- Available in EM0/EM1/EM2
- Available in EM0/EM1/EM2/EM3
- Available in EM0/EM1/EM2/EM3/EM4

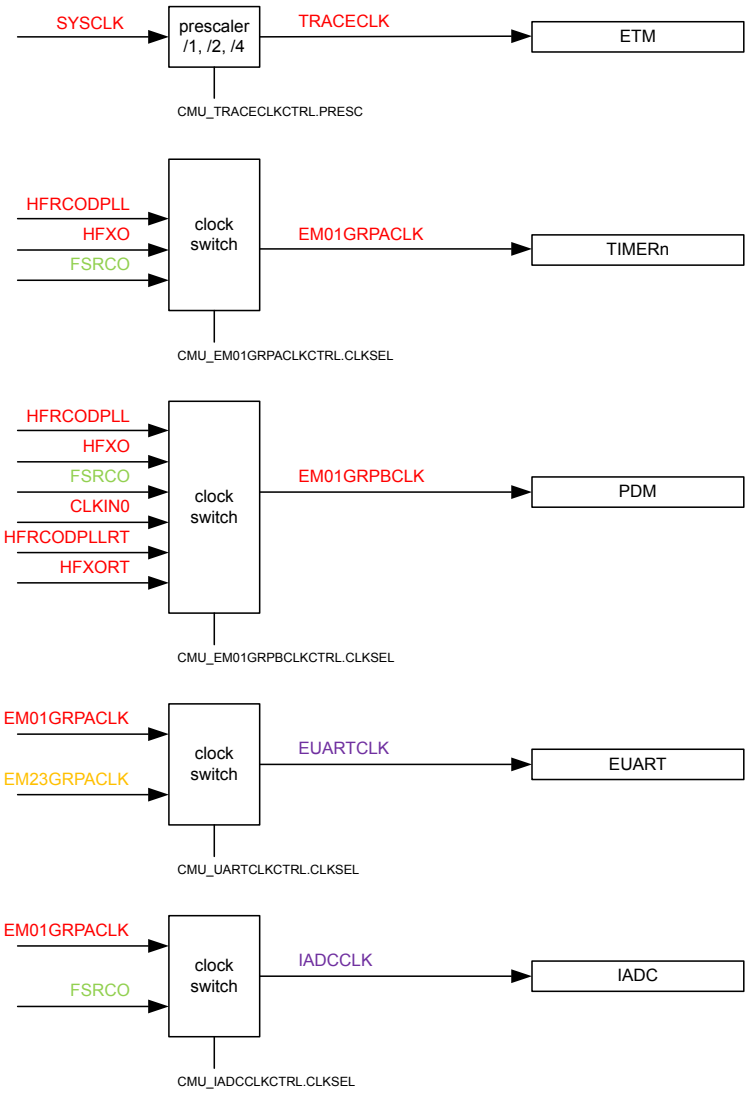


Figure 8.2. High Frequency Peripheral Clocks

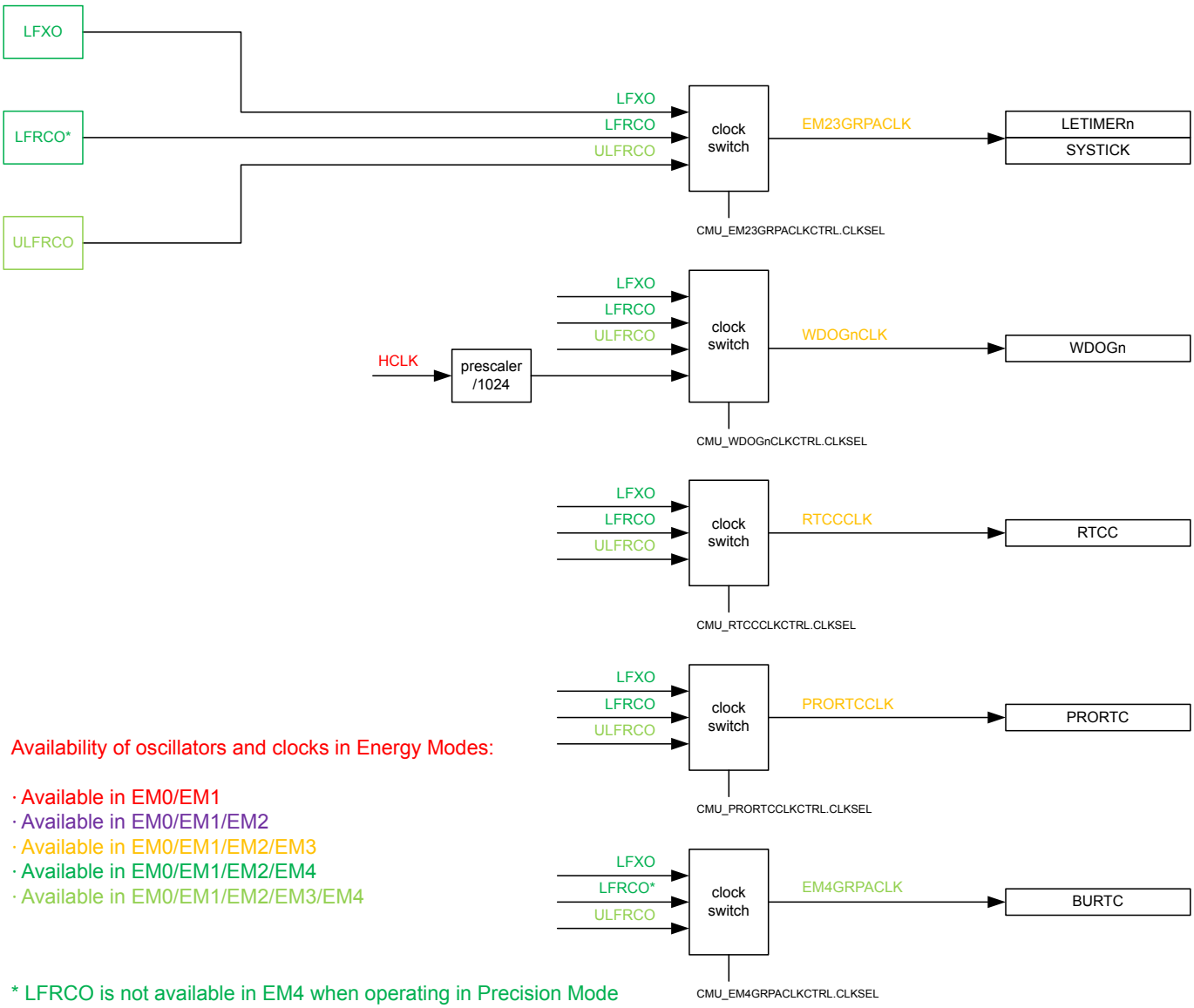


Figure 8.3. Low Frequency Peripheral Clocks

8.3.1 System Clocks

8.3.1.1 SYSCLK - Bus Clock

SYSCLK is the selected System Clock. HCLK is an optionally prescaled version of SYSCLK. PCLK is an optionally prescaled version of HCLK. The SYSCLK, and therefore HCLK and PCLK, can be driven by a high-frequency oscillator or be driven from a pin. By default, the FSRCO is selected as the bootup oscillator. To change the selected clock source, write to the CLKSEL bitfield in CMU_SYSCLKCTRL. If an invalid option is programmed into CLKSEL, FSRCO will be selected. The SYSCLK is running in EM0 Active and EM1 Sleep and is automatically stopped in EM2 DeepSleep.

The prescaler setting can be changed dynamically and the new setting takes effect immediately. When switching to a higher frequency oscillator source, prescaler setting should be adjusted before clock selection to prevent over clocking. For the same reason, when switching to a lower frequency oscillator source, prescaler setting cannot be adjusted until the clock selection is made.

The HFXO clock is fed directly to the Radio Transceiver. The clock received by the Radio Transceiver is therefore not affected by the selected clock source for SYSCLK nor by any clock prescaler.

8.3.1.2 HCLK - AHB Clock

HCLK is a prescaled version of SYSCLK. This clock drives the AHB bus interface. Example modules include the CPU, Cache, Bus Matrix, MSC, RAM, LDMA and GPIO. HCLK can be prescaled by setting HCLKPRESC in CMU_SYSCLKCTRL to DIV2 or DIV4. This prescales HCLK to all AHB bus clocks and is typically used to save energy in applications where the system is not required to run at the highest frequency. The setting can be changed dynamically and the new setting takes effect immediately. Some of the modules that are driven by this clock can be clock gated completely when not in use. This is done by clearing the module enable (EN) bit in the module's EN register.

8.3.1.3 PCLK - APB Clock

PCLK is a prescaled version of HCLK. This clock drives the APB bus interface. Example modules include USART and I2C. PCLK can be prescaled by setting PCLKPRESC in CMU_SYSCLKCTRL to DIV2. This prescales PCLK to all APB bus clocks and is necessary to prevent PCLK from exceeding the maximum frequency of 50 MHz. The setting can be changed dynamically and the new setting takes effect immediately. Some of the peripherals that are driven by this clock can be clock gated completely when not in use. This is done by clearing the module enable (EN) bit in the module's EN register.

8.3.1.4 LSPCLK - Low Speed APB Clock

LSPCLK is a prescaled version of PCLK. This clock drives the Low Speed APB bus interface. Example modules include I2C. LSPCLK is always prescaled by two. This prescales LSPCLK to all Low Speed APB bus clocks and is necessary to prevent LSPCLK from exceeding the maximum frequency of 25 MHz. Some of the peripherals that are driven by this clock can be clock gated completely when not in use. This is done by clearing the module enable (EN) bit in the module's EN register.

8.3.1.5 RHCLK - AHB Radio Clock

The radio AHB clock (RHCLK) is a prescaled version of SYSCLK. The maximum frequency for RHCLK is 50 MHz. The RHCLKPRESC setting in CMU_SYSCLKCTRL allows for SYSCLK to pass through (DIV1) or apply a divide-by-2 (DIV2) to the clock.

8.3.1.6 EM01GRPACLK - Energy Mode 01 Group A Clock

EM01GRPACLK is the selected clock for the Group A Peripherals operating in Energy Modes 0 or 1. These are typically high clock frequency peripheral modules. There are several selectable sources for EM01GRPACLK: HFXO, HFRCODPLL, and FSRCO. In addition, the EM01GRPACLK can be disabled. The selection is configured using the CLKSEL field in CMU_EM01GRPACLKCTRL.

Each High Frequency Peripheral that is clocked by EM01GRPACLK may have its own prescaler setting and enable bit. The prescaler settings, if available, can be found in the peripheral's control registers. The enable bit can be found in the module's EN register.

8.3.1.7 EM01GRPBCLK - Energy Mode 01 Group B Clock

EM01GRPBCLK is the selected clock for the Group B Peripherals operating in Energy Modes 0 or 1. These are typically high clock frequency peripheral modules. There are several selectable sources for EM01GRPBCLK: HFXO, HFRCODPLL, FSRCO, CLKIN0, HFRCODPLLRT, and HFXORT. In addition, the EM01GRPBCLK can be disabled. The selection is configured using the CLKSEL field in CMU_EM01GRPBCLKCTRL.

Each High Frequency Peripheral that is clocked by EM01GRPBCLK may have its own prescaler setting and enable bit. The prescaler settings, if available, can be found in the peripheral's control registers. The enable bit can be found in the module's EN register.

8.3.1.8 EM23GRPACLK - Energy Mode 2 and 3 Group A Clock

EM23GRPACLK is the selected clock for the Group A Peripherals operating down to Energy Modes 2 or 3. These are typically low energy consumption peripheral modules. There are three selectable sources for EM23GRPACLK: LFRCO, LFXO and ULFRCO. In addition, the EM23GRPACLK can be disabled. The selection is configured using the CLKSEL field in CMU_EM23GRPACLKCTRL.

Each Low Energy Peripheral that is clocked by EM23GRPACLK may have its own prescaler setting and enable bit. The prescaler settings, if available, can be found in the peripheral's control registers. The enable bit can be found in the module's EN register.

8.3.1.9 EM4GRPACLK - Energy Mode 4 Group A Clock

EM4GRPACLK is the selected clock for the Group A Peripherals operating down to Energy Mode 4. These are typically ultra low energy consumption peripheral modules. There are three selectable sources for EM4GRPACLK: LFRCO, LFXO and ULFRCO. In addition, the EM4GRPACLK can be disabled. The selection is configured using the CLKSEL field in CMU_EM4GRPACLKCTRL.

Note: EM4GRPACLK is in a different power domain than EM23GRPACLK, which makes it available all the way down to EM4.

Each Low Energy Peripheral that is clocked by EM4GRPACLK may have its own prescaler setting and enable bit. The prescaler settings, if available, can be found in the peripheral's control registers. The enable bit can be found in the module's EN register.

8.3.1.10 Peripheral Bus Clock Enable

Peripherals each have an individual bus clock enable bit in the CMU_CLKEN0 or CMU_CLKEN1 registers. Disabling the bus clock to a peripheral can save energy, even when that peripheral is not active.

8.3.1.11 IADCCLK - IADC Clock

IADCCLK is the selected clock for the IADC. The IADCCLK source may be selected from EM01GRPACLK or FSRCO. In addition, the IADCCLK can be disabled. The selection is configured using the CLKSEL field in CMU_IADCCLKCTRL.

Note: When using a Timer as the synchronous trigger for IADC conversion, EM01GRPACLK must be selected, because Timers run from EM01GRPACLK.

IADC has its own prescaler setting and enable bit. The prescaler settings can be found in the IADC's control registers. The enable bit can be found in the IADC's EN register.

Whichever clock source is selected as the IADC clock via the CLKSEL bitfield in the CMU_IADCCLKCTRL register, this clock will become active automatically when needed. The IADC can automatically start and stop it.

8.3.1.12 UARTCLK - UART Clock

UARTCLK is the selected clock for the EUART peripherals, and can choose between EM01GRPACLK and EM23GRPACLK. UARTCLK is selected via the CLKSEL field in CMU_UARTCLKCTRL. If operating in EM2, the EM23GRPACLK option must be chosen, as EM01GRPACLK will be shut down in these lower energy modes.

8.3.1.13 TRACECLK - Debug Trace Clock

The CMU scales the clock used for debug trace via the PRESC field in the CMU_TRACECLKCTRL register. The debug trace clock is limited to 50 MHz maximum. Therefore, if the SYSCLK is 50 MHz or less, the default DIV1 setting may be used. When SYSCLK is above 50 MHz, use DIV2 to avoid data pump overflow. The selected debug trace clock will be used to run the Cortex-M33 trace logic. Note that this register should be configured properly before enabling ETM.

8.3.1.14 WDOGCLK - Watchdog Timer Clock

The Watchdog Timer (WDOG) can be configured to use one of four different clock sources: LFRCO, LFXO, ULFRCO, or HCLKDIV1024. Select option HCLKDIV1024 to track Watchdog timeout with CPU clock speed.

8.3.2 Switching Clock Source

The FSRCO oscillator is a fixed frequency (20 MHz), low energy oscillator with extremely short start-up time. Therefore, this oscillator is chosen by hardware as the clock source for SYSCLK when the device starts up (e.g. after reset).

Software can switch between the different clock sources at run-time. For example, when the HFRCODPLL is the clock source, software can switch to HFXO by writing the field CLKSEL in the CMU_SYSCLKCTRL register. See Figure 8.4 CMU Switching from HFRCO to HFXO before HFXO is ready on page 140 for a description of the sequence of events for this specific operation.

When switching the SYSCLK to HFXO via the CLKSEL bitfield in CMU_SYSCLKCTRL, HFXO is automatically started. Switching to an oscillator that is not ready yet, the SYSCLK will stop for the duration of the oscillator start-up time. This effectively stalls the Core Modules. It is possible to avoid this by first enabling the target oscillator (e.g. HFXO) and then waiting for that oscillator to become ready before switching the clock source. This way, the system continues to run on the HFRCO until the target oscillator (e.g. HFXO) is ready and provides a reliable clock. This sequence of events is shown in Figure 8.5 CMU Switching from HFRCO to HFXO after HFXO is ready on page 141.

Generally, all oscillators have a separate flag that is set when the oscillator is ready. This flag can also be configured to generate an interrupt.

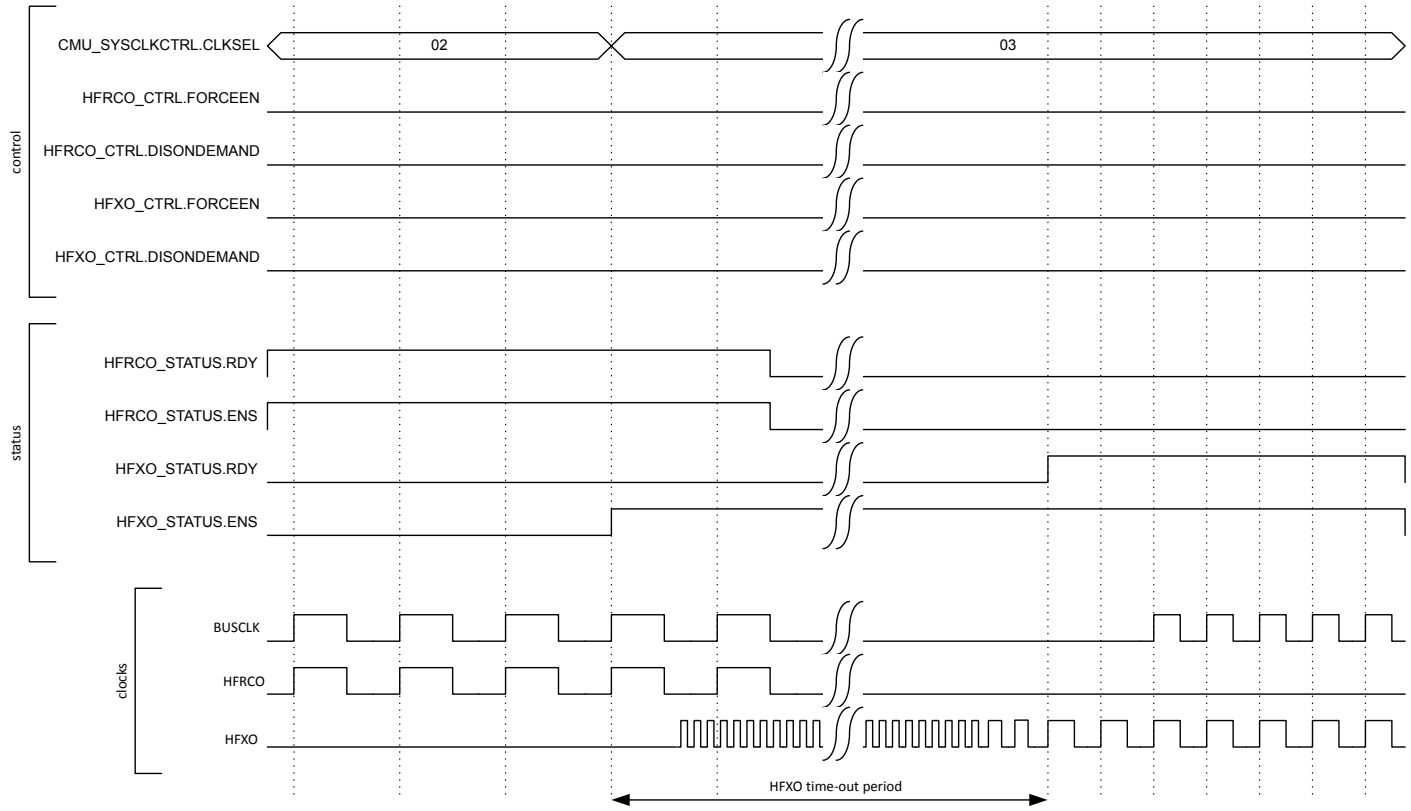


Figure 8.4. CMU Switching from HFRCO to HFXO before HFXO is ready

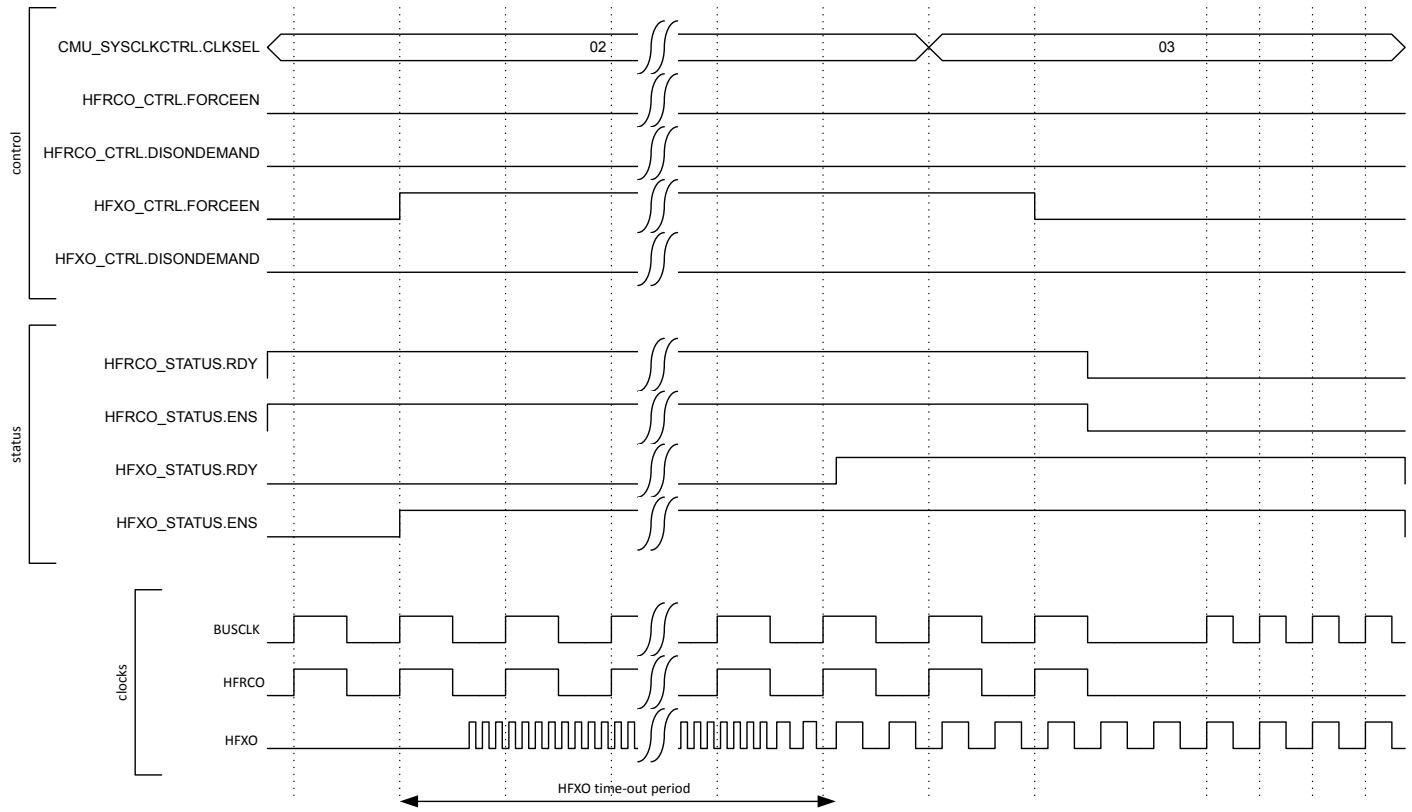


Figure 8.5. CMU Switching from HFRCO to HFXO after HFXO is ready

Switching clock source for various clock switches is done by setting the CLKSEL bitfields in CMU_*CLKCTRL. To ensure no stalls in the peripherals, the clock source should be ready before switching to it.

Note: To save energy, remember to disable all clock switches and/or module enable bits when not in use.

8.3.3 RC Oscillator Calibration

The CMU has built-in hardware support to efficiently calibrate RC oscillators (HFRCODPLL) at run-time, see Figure 8.6 Hardware Support for RC Oscillator Calibration on page 142 for an illustration of this circuit.

The concept is to select a reference and compare the RC frequency with the reference frequency. When the calibration circuit is started, one down-counter running on a selectable clock (DOWNSEL in CMU_CALCTRL) and one up-counter running on a selectable clock (UPSEL in CMU_CALCTRL) are started simultaneously. Reference clocks may also be routed through the PRS channels via the CALUP and CALDN consumer inputs. The top value for the down-counter must be written (CALTOP in CMU_CALCTRL) before calibration is started. The down-counter counts for CALTOP + 1 cycles. When the down-counter has reached 0, the up-counter is sampled and the CALRDY interrupt flag in the IF register is set. If CONT in CMU_CALCTRL is cleared, the counters are stopped after finishing the ongoing calibration. If continuous mode is selected by setting CONT in CMU_CALCTRL, the down-counter reloads the top value and continues counting, while the up-counter restarts from 0.

Software can then read out the sampled up-counter value from CMU_CALCNT. The up-counter has counted (the sampled value)+ 1 cycles. The ratio between the reference and the oscillator subject to the calibration can easily be found using (the top value)+1 and (the sampled value)+1. Overflows of the up-counter will not occur. If the up-counter reaches its top value before the down-counter reaches 0, the up-counter stays at its top value. Calibration can be started and stopped by writing CALSTART and CALSTOP bitfields in CMU_CALCMD, respectively. With this hardware support, it is simple to write efficient software calibration algorithms.

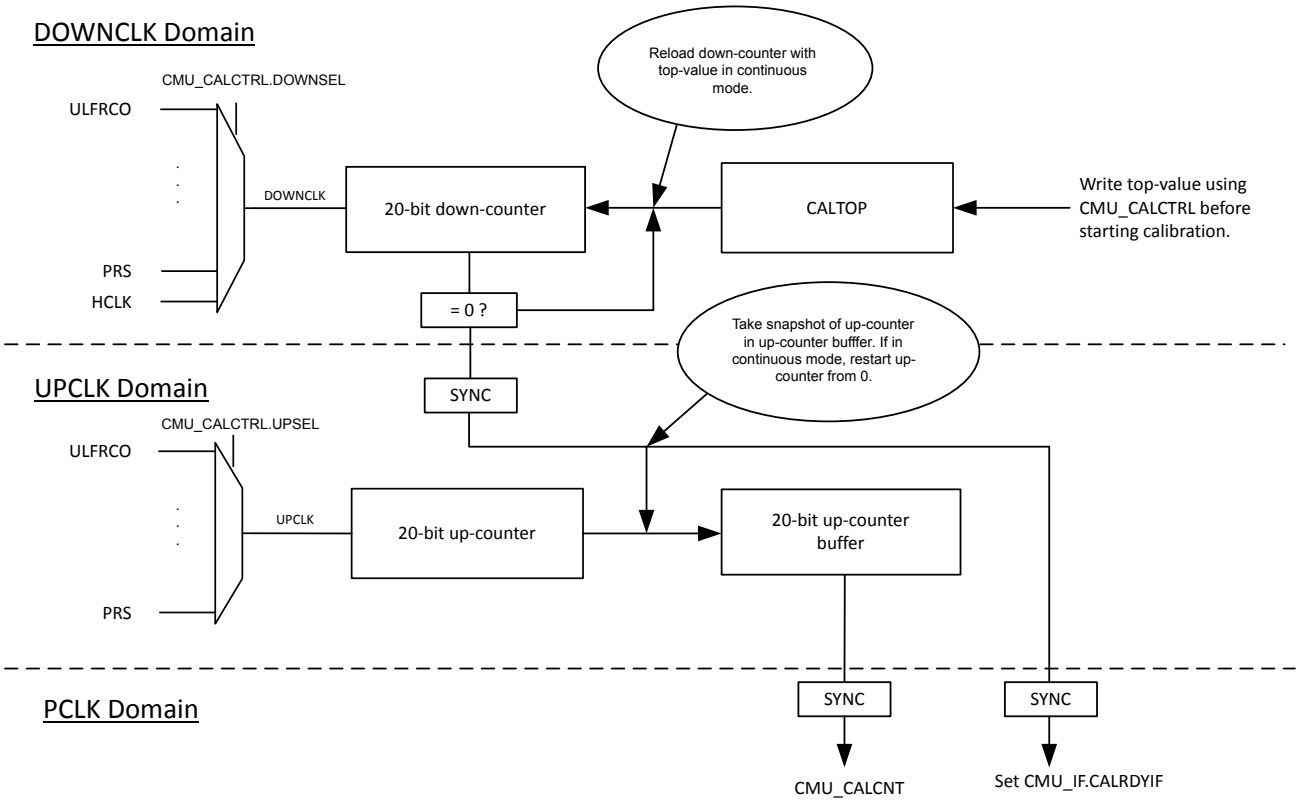


Figure 8.6. Hardware Support for RC Oscillator Calibration

The counter operation for single and continuous mode are shown in Figure 8.7 Single Calibration (CONT=0) on page 143 and Figure 8.8 Continuous Calibration (CONT=1) on page 144 respectively.

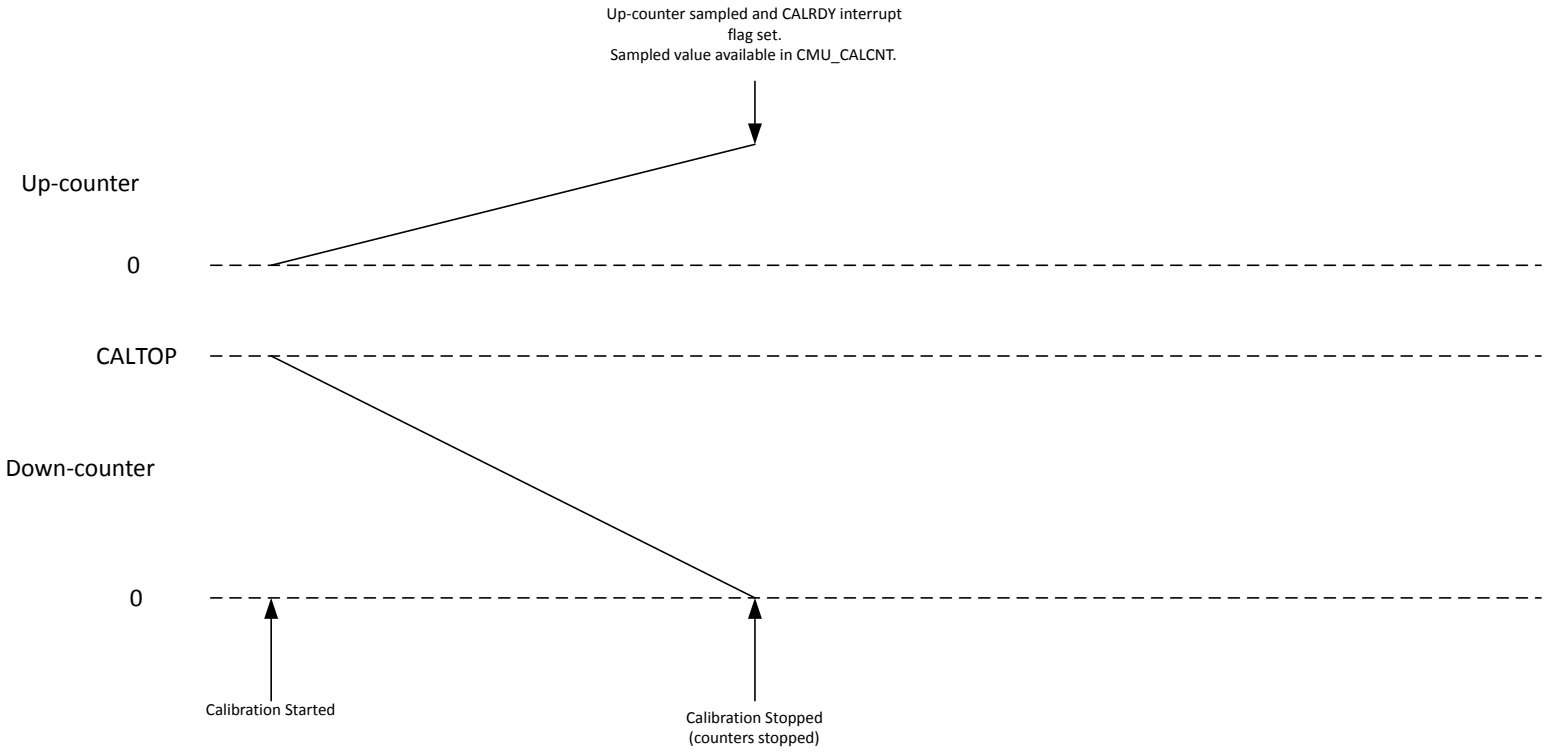


Figure 8.7. Single Calibration (CONT=0)

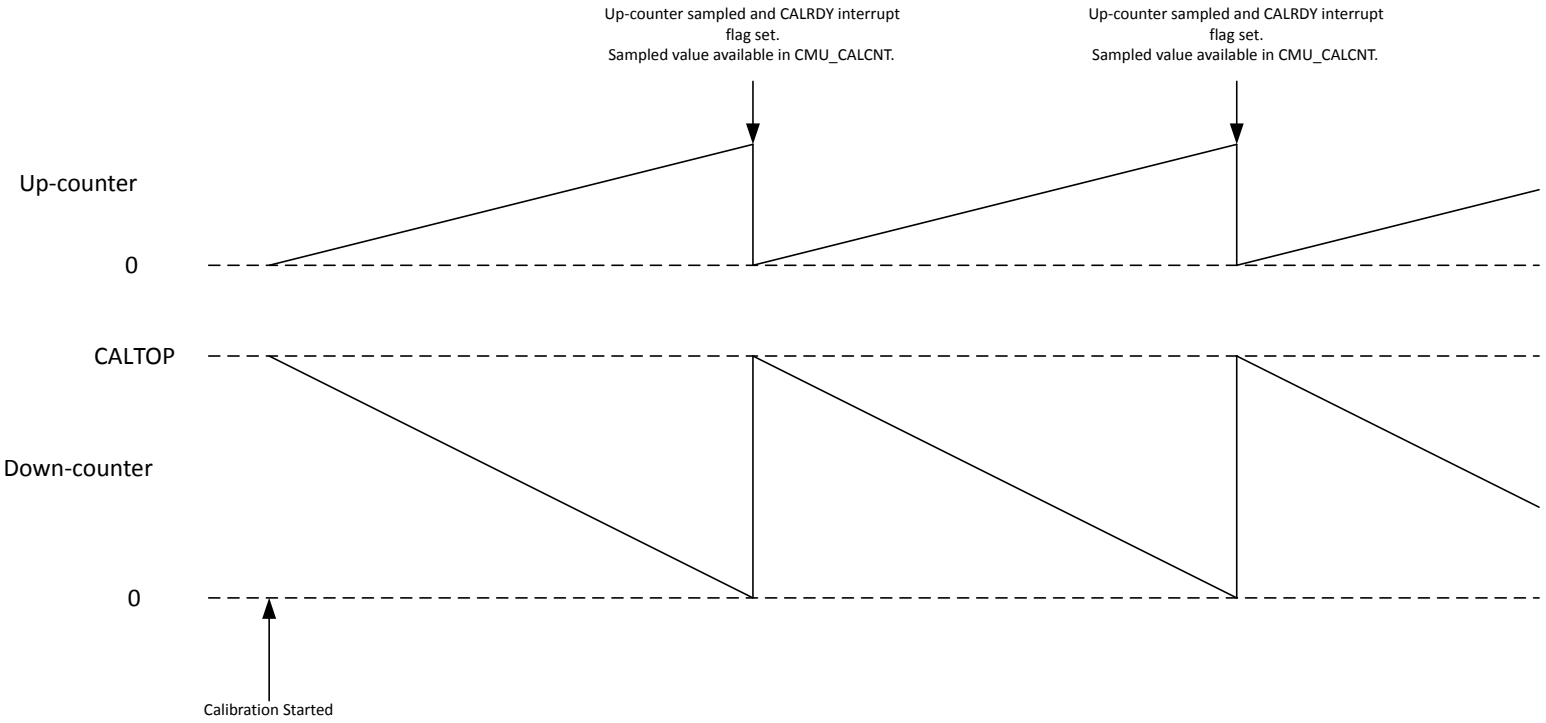


Figure 8.8. Continuous Calibration (CONT=1)

8.3.4 Energy Modes

The availability of oscillators and system clocks depends on the chosen energy mode. By default, the high frequency oscillators (HFRCODPLL and HFXO) and high frequency clocks (SYSCLK, HCLK, PCLK, RADIOCLK, and EM01GRPACLK) are available down to EM1 Sleep. From EM2 DeepSleep onwards these oscillators and clocks are normally off, although special cases exist as summarized in Table 8.1 Oscillator and clock availability in Energy Modes on page 145. The CMU overview figure in also indicates which oscillators and clocks can be used in what energy modes.

The low frequency oscillators (LFRCO and LFXO) are available in all energy modes except in EM3 Stop when they are off by definition. By default, these oscillators are also off in EM4 Shutoff. The LFXO or LFRCO (in non-precision mode) can be requested in EM4 as needed. When the LFRCO precision mode is used, this oscillator cannot operate in EM4. The ultra low frequency oscillator (ULFRCO) is on in all energy modes, except for EM4 Shutoff, but it can be requested on in that state as well if needed. The low frequency clocks (EM23GRPACLK, EM4GRPACLK, WDOGCLK, RTCCCLK, and PRORTCCLK) are in various power domains and therefore their availability not only depends on the chosen clock source, but also on the chosen energy mode as indicated in Table 8.1 Oscillator and clock availability in Energy Modes on page 145.

Table 8.1. Oscillator and clock availability in Energy Modes

	EM0 Active / EM1 Sleep	EM2 DeepSleep	EM3 Stop	EM4 Shutoff
HFRCODPLL	On [1]	Off	Off	Off
HFXO	On [1]	Off	Off	Off
FSRCO	On [1]	On [2]	On [2]	Off
LFRCO, LFXO	On [1]	On [1]	Off	On [3], [4]
ULFRCO	On	On	On	On [3]
SYSCLK, HCLK, PCLK, LSPCLK, RHCLK, EM01GRPACLK	On [1]	Off	Off	Off
IADCCLK	On [1]	On [2]	On [2]	Off
EM23GRPACLK, UARTCLK, WDOGCLK, RTCCCLK, PRORTCCLK	On [1]	On [1]	On [5]	Off
EM4GRPACLK	On [1]	On [1]	On [5]	On [3]

1 Under software control.
2 Default off, but kept active if used by the IADC.
3 Default off, but kept active if used by BURTC.
4 LFRCO not available in EM4 when precision mode enabled.
5 On only if ULFRCO is used as clock source.

8.3.5 Clock Output on a Pin

It is possible to configure the CMU to output clocks on the CMU_CLK pins. This clock selection is done using the CLKOUTSEL bitfields in CMU_EXPORTCLKCTRL. The required output pins must be enabled in the GPIO_DBUSCMU_ROUTEEN register and the pin locations can be configured in the GPIO_DBUSCMU_CLKOUT ROUTE register. The following clocks can be output on a pin:

- HCLK and EXPCLK. The HCLK is the high frequency clock for AHB. The EXPCLK is a prescaled version of HCLK as controlled by the PRESC bitfield in the CMU_EXPORTCLKCTRL register.
- The qualified clock from any of the oscillators. A qualified clock will not have any glitches or skewed duty-cycle during startup. For the LFXO and HFXO, correct configuration of the TIMEOUT bitfield(s) in LFXO_CFG and HFXO_XTALCFG, respectively is required to guarantee a properly qualified clock.

HCLK will not have a 50-50 duty cycle when any other division factor than 1 is used for HCLKPRESC bitfield in CMU_SYSCLKCTRL (i.e. if HCLKPRESC is not equal to 0). In such a case, the exported EXPCLK will also not be 50-50 when its division factor is not set to an even number in the PRESC bitfield of the CMU_EXPORTCLKCTRL register.

8.3.6 Clock Input from a Pin

It is possible to configure the CMU to input a clock from the CMU_CLKI0. This clock can be selected to drive SYSCLK and DPLL reference using CMU_SYSCLKCTRL.CLKSEL and CMU_DPLLREFCLKCTRL.CLKSEL respectively. The required input pin locations can be configured in the GPIO_DBUSCMU_CLKIN0ROUTE register.

8.3.7 Clock Output on PRS

The CMU can be used as a PRS producer. It can output clocks onto PRS which can be selected by a consumer as CMUCLKOUT. The clocks which can be produced via CMUCLKOUT are selected via the CLKOUTSEL fields in CMU_EXPORTCLKCTRL.

Note that the CLKOUTSEL fields are also used for selecting which clock is output onto a pin as described in 8.3.5 Clock Output on a Pin. In contrast with clock output on a pin however, output of a clock onto PRS does not depend on any configuration of the GPIO_DBUSCMU_ROUTEEN and GPIO_DBUSCMU_CLKOUT ROUTE registers.

8.3.8 Interrupts

The interrupts generated by the CMU module are combined into one interrupt vector. If CMU interrupts are enabled, an interrupt will be made if one or more of the interrupt flags in CMU_IF and their corresponding bits in CMU_IEN are set.

8.3.9 Protection

It is possible to lock the control and command registers to prevent unintended software writes to critical clock settings. This is controlled by the CMU_LOCK register.

The WDOGCLKCTRL registers are separately locked by CMU_WDOGLOCK register. This is to prevent EM3 Stop mode from disabling the watch dog clocks inadvertently.

In addition to software locks, hardware locks are implemented to prevent metastability. CMU_CALCTRL is locked by hardware when calibration is started by CMU_CALCMD.CALSTART. CMU_DPLLREFCLKCTRL is locked by hardware when DPLL is enabled via DPLL_EN.EN. Because these switches are not glitch-less, clock selection must be configured before enabling the operation and cannot be changed during operation.

8.4 CMU Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	CMU_IPVERSION	R	IP version ID
0x008	CMU_STATUS	RH	Status Register
0x010	CMU_LOCK	W	Configuration Lock Register
0x014	CMU_WDOGLOCK	W	WDOG Configuration Lock Register
0x020	CMU_IF	RWH INTFLAG	Interrupt Flag Register
0x024	CMU_IEN	RW	Interrupt Enable Register
0x050	CMU_CALCMD	W	Calibration Command Register
0x054	CMU_CALCTRL	RW	Calibration Control Register
0x058	CMU_CALCNT	R	Calibration Result Counter Register
0x064	CMU_CLKEN0	RW	Clock Enable Register 0
0x068	CMU_CLKEN1	RW	Clock Enable Register 1
0x070	CMU_SYSCLKCTRL	RW	System Clock Control
0x080	CMU_TRACECLKCTRL	RW	Debug Trace Clock Control
0x090	CMU_EXPORTCLKCTRL	RW	Export Clock Control
0x100	CMU_DPLLREFCLKCTRL	RW	Digital PLL Reference Clock Control
0x120	CMU_EM01GRPACLKCTRL	RW	EM01 Peripheral Group A Clock Control
0x124	CMU_EM01GRPBCLKCTRL	RW	EM01 Peripheral Group B Clock Control
0x140	CMU_EM23GRPACLKCTRL	RW	EM23 Peripheral Group A Clock Control
0x160	CMU_EM4GRPACLKCTRL	RW	EM4 Peripheral Group A Clock Control
0x180	CMU_IADCCLKCTRL	RW	IADC Clock Control
0x200	CMU_WDOG0CLKCTRL	RW	Watchdog0 Clock Control
0x220	CMU_EUART0CLKCTRL	RW	UART Clock Control
0x240	CMU_RTCCCLKCTRL	RW	RTCC Clock Control
0x260	CMU_CRYPTOACCCLKCTRL	RW	CRYPTOACC Clock Control
0x280	CMU_RADIOCLKCTRL	RW	Radio Clock Control
0x1000	CMU_IPVERSION_SET	R	IP version ID
0x1008	CMU_STATUS_SET	RH	Status Register
0x1010	CMU_LOCK_SET	W	Configuration Lock Register
0x1014	CMU_WDOGLOCK_SET	W	WDOG Configuration Lock Register
0x1020	CMU_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1024	CMU_IEN_SET	RW	Interrupt Enable Register
0x1050	CMU_CALCMD_SET	W	Calibration Command Register
0x1054	CMU_CALCTRL_SET	RW	Calibration Control Register
0x1058	CMU_CALCNT_SET	R	Calibration Result Counter Register
0x1064	CMU_CLKEN0_SET	RW	Clock Enable Register 0

Offset	Name	Type	Description
0x1068	CMU_CLKEN1_SET	RW	Clock Enable Register 1
0x1070	CMU_SYSCLKCTRL_SET	RW	System Clock Control
0x1080	CMU_TRACECLKCTRL_SET	RW	Debug Trace Clock Control
0x1090	CMU_EXPORTCLKCTRL_SET	RW	Export Clock Control
0x1100	CMU_DPLLREFCLKCTRL_SET	RW	Digital PLL Reference Clock Control
0x1120	CMU_EM01GRPACLKCTRL_SET	RW	EM01 Peripheral Group A Clock Control
0x1124	CMU_EM01GRPBCLKCTRL_SET	RW	EM01 Peripheral Group B Clock Control
0x1140	CMU_EM23GRPACLKCTRL_SET	RW	EM23 Peripheral Group A Clock Control
0x1160	CMU_EM4GRPACLKCTRL_SET	RW	EM4 Peripheral Group A Clock Control
0x1180	CMU_IADCCLKCTRL_SET	RW	IADC Clock Control
0x1200	CMU_WDOG0CLKCTRL_SET	RW	Watchdog0 Clock Control
0x1220	CMU_EUART0CLKCTRL_SET	RW	UART Clock Control
0x1240	CMU_RTCCCLKCTRL_SET	RW	RTCC Clock Control
0x1260	CMU_CRYPTOACCCLKCTRL_SET	RW	CRYPTOACC Clock Control
0x1280	CMU_RADIOCLKCTRL_SET	RW	Radio Clock Control
0x2000	CMU_IPVERSION_CLR	R	IP version ID
0x2008	CMU_STATUS_CLR	RH	Status Register
0x2010	CMU_LOCK_CLR	W	Configuration Lock Register
0x2014	CMU_WDOGLOCK_CLR	W	WDOG Configuration Lock Register
0x2020	CMU_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2024	CMU_IEN_CLR	RW	Interrupt Enable Register
0x2050	CMU_CALCMD_CLR	W	Calibration Command Register
0x2054	CMU_CALCTRL_CLR	RW	Calibration Control Register
0x2058	CMU_CALCNT_CLR	R	Calibration Result Counter Register
0x2064	CMU_CLKEN0_CLR	RW	Clock Enable Register 0
0x2068	CMU_CLKEN1_CLR	RW	Clock Enable Register 1
0x2070	CMU_SYSCLKCTRL_CLR	RW	System Clock Control
0x2080	CMU_TRACECLKCTRL_CLR	RW	Debug Trace Clock Control
0x2090	CMU_EXPORTCLKCTRL_CLR	RW	Export Clock Control
0x2100	CMU_DPLLREFCLKCTRL_CLR	RW	Digital PLL Reference Clock Control
0x2120	CMU_EM01GRPACLKCTRL_CLR	RW	EM01 Peripheral Group A Clock Control
0x2124	CMU_EM01GRPBCLKCTRL_CLR	RW	EM01 Peripheral Group B Clock Control
0x2140	CMU_EM23GRPACLKCTRL_CLR	RW	EM23 Peripheral Group A Clock Control

Offset	Name	Type	Description
0x2160	CMU_EM4GRPACLKCTRL_CLR	RW	EM4 Peripheral Group A Clock Control
0x2180	CMU_IADCCLKCTRL_CLR	RW	IADC Clock Control
0x2200	CMU_WDOG0CLKCTRL_CLR	RW	Watchdog0 Clock Control
0x2220	CMU_EUART0CLKCTRL_CLR	RW	UART Clock Control
0x2240	CMU_RTCCCLKCTRL_CLR	RW	RTCC Clock Control
0x2260	CMU_CRYP-TOACCCLKCTRL_CLR	RW	CRYPTOACC Clock Control
0x2280	CMU_RADIOCLKCTRL_CLR	RW	Radio Clock Control
0x3000	CMU_IPVERSION_TGL	R	IP version ID
0x3008	CMU_STATUS_TGL	RH	Status Register
0x3010	CMU_LOCK_TGL	W	Configuration Lock Register
0x3014	CMU_WDOGLOCK_TGL	W	WDOG Configuration Lock Register
0x3020	CMU_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3024	CMU_IEN_TGL	RW	Interrupt Enable Register
0x3050	CMU_CALCMD_TGL	W	Calibration Command Register
0x3054	CMU_CALCTRL_TGL	RW	Calibration Control Register
0x3058	CMU_CALCNT_TGL	R	Calibration Result Counter Register
0x3064	CMU_CLKEN0_TGL	RW	Clock Enable Register 0
0x3068	CMU_CLKEN1_TGL	RW	Clock Enable Register 1
0x3070	CMU_SYSCLKCTRL_TGL	RW	System Clock Control
0x3080	CMU_TRACECLKCTRL_TGL	RW	Debug Trace Clock Control
0x3090	CMU_EXPORTCLKCTRL_TGL	RW	Export Clock Control
0x3100	CMU_DPLLREFCLKCTRL_TGL	RW	Digital PLL Reference Clock Control
0x3120	CMU_EM01GRPACLKCTRL_TGL	RW	EM01 Peripheral Group A Clock Control
0x3124	CMU_EM01GRPBCLKCTRL_TGL	RW	EM01 Peripheral Group B Clock Control
0x3140	CMU_EM23GRPACLKCTRL_TGL	RW	EM23 Peripheral Group A Clock Control
0x3160	CMU_EM4GRPACLKCTRL_TGL	RW	EM4 Peripheral Group A Clock Control
0x3180	CMU_IADCCLKCTRL_TGL	RW	IADC Clock Control
0x3200	CMU_WDOG0CLKCTRL_TGL	RW	Watchdog0 Clock Control
0x3220	CMU_EUART0CLKCTRL_TGL	RW	UART Clock Control
0x3240	CMU_RTCCCLKCTRL_TGL	RW	RTCC Clock Control
0x3260	CMU_CRYP-TOACCCLKCTRL_TGL	RW	CRYPTOACC Clock Control
0x3280	CMU_RADIOCLKCTRL_TGL	RW	Radio Clock Control

8.5 CMU Register Description

8.5.1 CMU_IPVERSION - IP version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x1	R	**IP Version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

8.5.2 CMU_STATUS - Status Register

Offset	Bit Position																														
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0																														0x0
Access	R	R																														R
Name	LOCK	WDOGLOCK																														CALRDY

Bit	Name	Reset	Access	Description
31	LOCK	0x0	R	**Configuration Lock Status**
	Indicates the current status of configuration lock			
	Value	Mode		Description
	0	UNLOCKED		Configuration lock is unlocked
	1	LOCKED		Configuration lock is locked
30	WDOGLOCK	0x0	R	**Configuration Lock Status for WDOG**
	Indicates the current status of WDOG configuration lock			
	Value	Mode		Description
	0	UNLOCKED		WDOG configuration lock is unlocked
	1	LOCKED		WDOG configuration lock is locked
29:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CALRDY	0x0	R	**Calibration Ready**
	Calibration is Ready (0 when calibration is ongoing).			

8.5.3 CMU_LOCK - Configuration Lock Register

Offset	Bit Position																															
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x93F7															
Access																	W															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x93F7	W	**Configuration Lock Key**

Write any other value than the unlock code to lock registers from editing. Write the unlock code to unlock.

Value	Mode	Description
37879	UNLOCK	Write this value to unlock

8.5.4 CMU_WDOGLOCK - WDOG Configuration Lock Register

Offset	Bit Position																															
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x5257															
Access																	W															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x5257	W	**Configuration Lock Key**

Write any other value than the unlock code to lock registers from editing. Write the unlock code to unlock.

Value	Mode	Description
37879	UNLOCK	Write this value to unlock

8.5.5 CMU_IF - Interrupt Flag Register

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															CALOF	CALRDY

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	CALOF	0x0	RW	**Calibration Overflow Interrupt Flag**
	Set when calibration overflow has occurred (i.e. if a new calibration completes before CMU_CALSTATUS has been read)			
0	CALRDY	0x0	RW	**Calibration Ready Interrupt Flag**
	Set when calibration is completed			

8.5.6 CMU_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															CALOF	CALRDY

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	CALOF	0x0	RW	**Calibration Overflow Interrupt Enable**
	Enable/disable CALOF interrupt			
0	CALRDY	0x0	RW	**Calibration Ready Interrupt Enable**
	Enable/disable CALRDY interrupt			

8.5.7 CMU_CALCMD - Calibration Command Register

Offset	Bit Position																													
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															W	W
Name																															CALSTOP	CALSTART

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	CALSTOP	0x0	W	**Calibration Stop**
	Stops the calibration counters.			
0	CALSTART	0x0	W	**Calibration Start**
	Starts the calibration, effectively loading the CMU_CALCTRL.CALCNT into the down-counter and start decrementing.			

8.5.8 CMU_CALCTRL - Calibration Control Register

Offset	Bit Position																															
0x054	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0				0x0				0x0				0x0																			
Access	RW				RW				RW				RW																			
Name	DOWNSEL				UPSEL				CONT				CALTOP																			

Bit	Name	Reset	Access	Description
31:28	DOWNSEL	0x0	RW	**Calibration Down-counter Select**

Selects clock source for the calibration down-counter. Changing this while calibration is running results in bus fault..

Value	Mode	Description
0	DISABLED	Down-counter is not clocked
1	HCLK	HCLK is clocking down-counter
2	PRS	PRS CMU_CALDN consumer is clocking down-counter
3	HFXO	HFXO is clocking down-counter
4	LFXO	LFXO is clocking down-counter
5	HFRCODPLL	HFRCODPLL is clocking down-counter
9	FSRCO	FSRCO is clocking down-counter
10	LFRCO	LFRCO is clocking down-counter
11	ULFRCO	ULFRCO is clocking down-counter

Bit	Name	Reset	Access	Description
27:24	UPSEL	0x0	RW	**Calibration Up-counter Select**

Selects clock source for the calibration up-counter. Changing this while calibration is running results in bus fault.

Value	Mode	Description
0	DISABLED	Up-counter is not clocked
1	PRS	PRS CMU_CALUP consumer is clocking up-counter
2	HFXO	HFXO is clocking up-counter
3	LFXO	LFXO is clocking up-counter
4	HFRCODPLL	HFRCODPLL is clocking up-counter
8	FSRCO	FSRCO is clocking up-counter
9	LFRCO	LFRCO is clocking up-counter
10	ULFRCO	ULFRCO is clocking up-counter

Bit	Name	Reset	Access	Description
23	CONT	0x0	RW	**Continuous Calibration**

Set this bit to enable continuous calibration. Changing this while calibration is running results in bus fault.

Bit	Name	Description
22:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*

Bit	Name	Reset	Access	Description
19:0	CALTOP	0x0	RW	**Calibration Counter Top Value**
	Write top value before calibration. Changing this while calibration is running results in bus fault.			

8.5.9 CMU_CALCNT - Calibration Result Counter Register

Offset	Bit Position																															
0x058	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0									
Access																							R									
Name																							CALCNT									

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:0	CALCNT	0x0	R	**Calibration Result Counter Value**
	Read calibration result when Calibration Ready flag has been set.			

8.5.10 CMU_CLKEN0 - Clock Enable Register 0

Offset	Bit Position																													
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name	DCDC	RTCC	BURTC	BURAM	PRS	GPIO	PDM	EUART0	ULFRCO	LFXO	LFRCO	FSRCO	HFXO0	HFRCO0	DPLL0	SYSCFG	I2C1	I2C0	WDOG0	LETIMER0	AMUXCP0	IADC0	USART1	USART0	TIMER3	TIMER2	TIMER1	TIMER0	GPCRC	RADIOAES	LDMAXBAR	LDMA

Bit	Name	Reset	Access	Description
31	DCDC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
30	RTCC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
29	BURTC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
28	BURAM	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
27	PRS	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
26	GPIO	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
25	PDM	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
24	EUART0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
23	ULFRCO	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
22	LFXO	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
21	LFRCO	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
20	FSRCO	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
19	HFXO0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
18	HFRCO0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			

Bit	Name	Reset	Access	Description
17	DPLL0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
16	SYSCFG	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
15	I2C1	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
14	I2C0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
13	WDOG0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
12	LETIMER0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
11	AMUXCP0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
10	IADC0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
9	USART1	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
8	USART0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
7	TIMER3	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
6	TIMER2	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
5	TIMER1	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
4	TIMER0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
3	GPCRC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
2	RADIOAES	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
1	LDMAXBAR	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
0	LDMA	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			

8.5.11 CMU_CLKEN1 - Clock Enable Register 1

Offset	Bit Position																													
0x068	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access														RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name														TIMER4	MSC	ICACHE0	SMU	RFSENSE	CRYPTOACC	IFADCDEBUG	BUFC	PRORTC	RDMAILBOX1	RDMAILBOX0	RDSCRATCHPAD	SYNTH	RAC	PROTIMER	FRC	RFCRC	MODEM	AGC

Bit	Name	Reset	Access	Description
31:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	TIMER4	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
17	MSC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
16	ICACHE0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
15	SMU	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
14	RFSENSE	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
13	CRYPTOACC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
12	IFADCDEBUG	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
11	BUFC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
10	PRORTC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
9	RDMAILBOX1	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
8	RDMAILBOX0	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
7	RDSCRATCHPAD	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
6	SYNTH	0x0	RW	**Enable Bus Clock**

Bit	Name	Reset	Access	Description
	Enables module PCLK/HCLK			
5	RAC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
4	PROTIMER	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
3	FRC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
2	RFCRC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
1	MODEM	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			
0	AGC	0x0	RW	**Enable Bus Clock**
	Enables module PCLK/HCLK			

8.5.12 CMU_SYSCLKCTRL - System Clock Control

Offset	Bit Position																															
0x070	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0	0x0					0x0									0x1	
Access																RW	RW					RW									RW	
Name																RHCLKPRESC	HCLKPRESC					PCLKPRESC									CLKSEL	

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	RHCLKPRESC	0x0	RW	**Radio HCLK Prescaler**
	Specifies the clock divider for Radio HCLK			

Value	Mode	Description
0	DIV1	Radio HCLK is SYSCLK divided by 1
1	DIV2	Radio HCLK is SYSCLK divided by 2

Bit	Name	Reset	Access	Description
15:12	HCLKPRESC	0x0	RW	**HCLK Prescaler**
	Specifies the clock divider for HCLK			

Value	Mode	Description
0	DIV1	HCLK is SYSCLK divided by 1
1	DIV2	HCLK is SYSCLK divided by 2
3	DIV4	HCLK is SYSCLK divided by 4
7	DIV8	HCLK is SYSCLK divided by 8
15	DIV16	HCLK is SYSCLK divided by 16

Bit	Name	Reset	Access	Description
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	PCLKPRESC	0x0	RW	**PCLK Prescaler**
	Specifies the clock divider for PCLK			

Value	Mode	Description
0	DIV1	PCLK is HCLK divided by 1
1	DIV2	PCLK is HCLK divided by 2

Bit	Name	Reset	Access	Description
9:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	CLKSEL	0x1	RW	**Clock Select**
	Selects the clock source for SYSCLK.			

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	1	FSRCO		FSRCO is clocking SYSCLK
	2	HFRCODPLL		HFRCODPLL is clocking SYSCLK
	3	HFXO		HFXO is clocking SYSCLK
	4	CLKIN0		CLKIN0 is clocking SYSCLK

8.5.13 CMU_TRACECLKCTRL - Debug Trace Clock Control

Offset	Bit Position																															
0x080	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0					
Access																											RW					
Name																											PRESC					

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	PRESC	0x0	RW	**TRACECLK Prescaler**
	Clock prescaler for debug trace logic.			
	Value	Mode		Description
	0	DIV1		TRACECLK is SYSCLK divided by 1
	1	DIV2		TRACECLK is SYSCLK divided by 2
	3	DIV4		TRACECLK is SYSCLK divided by 4
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

8.5.14 CMU_EXPORTCLKCTRL - Export Clock Control

Offset	Bit Position																													
0x090	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0									0x0								0x0								0x0			
Access				RW									RW								RW								RW			
Name				PRESC									CLKOUTSEL2								CLKOUTSEL1								CLKOUTSEL0			

Bit	Name	Reset	Access	Description
31:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28:24	PRESC	0x0	RW	**EXPORTCLK Prescaler**
	Specifies the clock divider for EXPORTCLK (relative to SYSCLK).			
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	CLKOUTSEL2	0x0	RW	**Clock Output Select 2**
	Controls the clock output 2 multiplexer.			

Value	Mode	Description
0	DISABLED	CLKOUT2 is not clocked
1	HCLK	HCLK is clocking CLKOUT2
2	HFEXPCLK	HFEXPCLK is clocking CLKOUT2
3	ULFRCO	ULFRCO is clocking CLKOUT2
4	LFRCO	LFRCO is clocking CLKOUT2
5	LFXO	LFXO is clocking CLKOUT2
6	HFRCODPLL	HFRCODPLL is clocking CLKOUT2
7	HFXO	HFXO is clocking CLKOUT2
8	FSRCO	FSRCO is clocking CLKOUT2

15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	CLKOUTSEL1	0x0	RW	**Clock Output Select 1**
	Controls the clock output 1 multiplexer.			

Value	Mode	Description
0	DISABLED	CLKOUT1 is not clocked
1	HCLK	HCLK is clocking CLKOUT1
2	HFEXPCLK	HFEXPCLK is clocking CLKOUT1
3	ULFRCO	ULFRCO is clocking CLKOUT1

Bit	Name	Reset	Access	Description
	4	LFRCO		LFRCO is clocking CLKOUT1
	5	LFXO		LFXO is clocking CLKOUT1
	6	HFRCODPLL		HFRCODPLL is clocking CLKOUT1
	7	HFXO		HFXO is clocking CLKOUT1
	8	FSRCO		FSRCO is clocking CLKOUT1
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	CLKOUTSEL0	0x0	RW	**Clock Output Select 0**
	Controls the clock output 0 multiplexer.			
	Value	Mode		Description
	0	DISABLED		CLKOUT0 is not clocked
	1	HCLK		HCLK is clocking CLKOUT0
	2	HFEXPCLK		HFEXPCLK is clocking CLKOUT0
	3	ULFRCO		ULFRCO is clocking CLKOUT0
	4	LFRCO		LFRCO is clocking CLKOUT0
	5	LFXO		LFXO is clocking CLKOUT0
	6	HFRCODPLL		HFRCODPLL is clocking CLKOUT0
	7	HFXO		HFXO is clocking CLKOUT0
	8	FSRCO		FSRCO is clocking CLKOUT0

8.5.15 CMU_DPLLREFCLKCTRL - Digital PLL Reference Clock Control

Offset	Bit Position																													
0x100	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	
Access																															RW	
Name																															CLKSEL	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	CLKSEL	0x0	RW	**Clock Select**

Selects the clock source for DPLL reference. Changing this while DPLL is enabled results in bus fault.

Value	Mode	Description
0	DISABLED	DPLLREFCLK is not clocked
1	HFXO	HFXO is clocking DPLLREFCLK
2	LFXO	LFXO is clocking DPLLREFCLK
3	CLKIN0	CLKIN0 is clocking DPLLREFCLK

8.5.16 CMU_EM01GRPACLKCTRL - EM01 Peripheral Group A Clock Control

Offset	Bit Position																													
0x120	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x1	
Access																															RW	
Name																															CLKSEL	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	CLKSEL	0x1	RW	**Clock Select**

Selects the clock source for EM01 Group A Clock.

Value	Mode	Description
1	HFRCODPLL	HFRCODPLL is clocking EM01GRPACLK
2	HFXO	HFXO is clocking EM01GRPACLK
3	FSRCO	FSRCO is clocking EM01GRPACLK

8.5.17 CMU_EM01GRPBCLKCTRL - EM01 Peripheral Group B Clock Control

Offset	Bit Position																														
0x124	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x1		
Access																														RW		
Name																														CLKSEL		

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	CLKSEL	0x1	RW	**Clock Select**

Selects the clock source for EM01 Group B Clock.

Value	Mode	Description
1	HFRCODPLL	HFRCODPLL is clocking EM01GRPBCLK
2	HFXO	HFXO is clocking EM01GRPBCLK
3	FSRCO	FSRCO is clocking EM01GRPBCLK
4	CLKIN0	CLKIN0 is clocking EM01GRPBCLK
5	HFRCODPLLRT	HFRCODPLL (re-timed) is clocking EM01GRPBCLK
6	HFXORT	HFXO (re-timed) is clocking EM01GRPBCLK

8.5.18 CMU_EM23GRPACLKCTRL - EM23 Peripheral Group A Clock Control

Offset	Bit Position																													
0x140	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x1	
Access																															RW	
Name																															CLKSEL	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	CLKSEL	0x1	RW	**Clock Select**

Selects the clock source for EM23 Group A Clock.

Value	Mode	Description
1	LFRCO	LFRCO is clocking EM23GRPACLK
2	LFXO	LFXO is clocking EM23GRPACLK
3	ULFRCO	ULFRCO is clocking EM23GRPACLK

8.5.19 CMU_EM4GRPACLKCTRL - EM4 Peripheral Group A Clock Control

Offset	Bit Position																													
0x160	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x1	
Access																															RW	
Name																															CLKSEL	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	CLKSEL	0x1	RW	**Clock Select**

Selects the clock source for EM4 Group A Clock.

Value	Mode	Description
1	LFRCO	LFRCO is clocking EM4GRPACLK
2	LFXO	LFXO is clocking EM4GRPACLK
3	ULFRCO	ULFRCO is clocking EM4GRPACLK

8.5.20 CMU_IADCCLKCTRL - IADC Clock Control

Offset	Bit Position																													
0x180	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x1	
Access																															RW	
Name																															CLKSEL	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	CLKSEL	0x1	RW	**Clock Select**

Selects the clock source for for IADC. EM01GRPACLK should never be selected as clock source for IADC when disabling the EM01GRACLK (e.g. because of EM23 entry).

Value	Mode	Description
1	EM01GRPACLK	EM01GRPACLK is clocking IADCCLK
2	FSRCO	FSRCO is clocking IADCCLK

8.5.21 CMU_WDOG0CLKCTRL - Watchdog0 Clock Control

Offset	Bit Position																													
0x200	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x1	
Access																															RW	
Name																															CLKSEL	

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	CLKSEL	0x1	RW	**Clock Select**

Selects the clock source for WDOG0.

Value	Mode	Description
1	LFRCO	LFRCO is clocking WDOG0CLK
2	LFXO	LFXO is clocking WDOG0CLK
3	ULFRCO	ULFRCO is clocking WDOG0CLK
4	HCLKDIV1024	HCLKDIV1024 is clocking WDOG0CLK

8.5.22 CMU_EUART0CLKCTRL - UART Clock Control

Offset	Bit Position																													
0x220	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x1	
Access																															RW	
Name																															CLKSEL	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	CLKSEL	0x1	RW	**Clock Select**

This bit controls which clock is used for UART. EM01GRPACLK should never be selected as clock source for UART when disabling the EM01GRACLK (e.g. because of EM23 entry).

Value	Mode	Description
0	DISABLED	UART is not clocked
1	EM01GRPACLK	EM01GRPACLK is clocking UART
2	EM23GRPACLK	EM23GRPACLK is clocking UART

8.5.23 CMU_RTCCCLKCTRL - RTCC Clock Control

Offset	Bit Position																													
0x240	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x1	
Access																															RW	
Name																															CLKSEL	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	CLKSEL	0x1	RW	**Clock Select**

Selects the clock source for RTCC.

Value	Mode	Description
1	LFRCO	LFRCO is clocking RTCCCLK
2	LFXO	LFXO is clocking RTCCCLK
3	ULFRCO	ULFRCO is clocking RTCCCLK

8.5.24 CMU_CRYPTOACCCLKCTRL - CRYPTOACC Clock Control

Offset	Bit Position																													
0x260	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															AESEN	PKEN

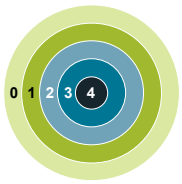
Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	AESEN	0x0	RW	**AES Enable**
	Enables clock to the AES sub-module			
0	PKEN	0x0	RW	**PK Enable**
	Enables clock to the PK sub-module			

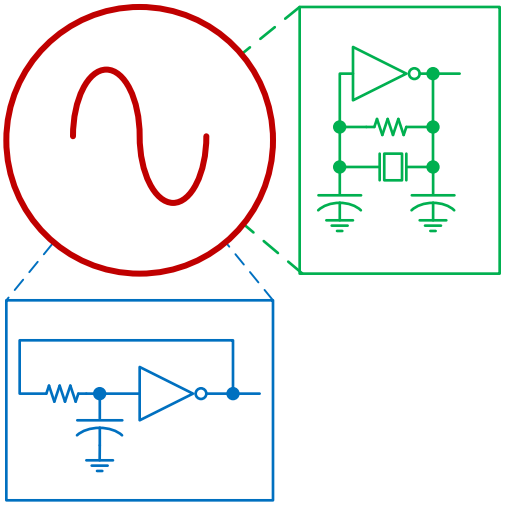
8.5.25 CMU_RADIOCLKCTRL - Radio Clock Control

Offset	Bit Position																													
0x280	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															0x0
Access	RW																															RW
Name	DBGCLK																															EN

Bit	Name	Reset	Access	Description
31	DBGCLK	0x0	RW	**Enable Clock for Debugger**
	When set to 1, this forces radio busmatrix and RAC clocks to run, allowing RAC sequencer debugger to stay attached.			
30:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**Enable**
	Enables radio bridge clock			

9. Oscillators





Quick Facts

What?

The EFR32xG22 has a wide range of high frequency and low frequency oscillators.

Why?

The High Frequency oscillators support EM0/1 operation. The Low-frequency oscillators provide a low frequency clock for the low energy peripherals in EM/2/3/4.

How?

The HFXO supports high freqeuency crystal oscillators. The LFXO supports 32.768 kHz crystal oscillators. The RC oscillators are internal oscillators that require no external components.

9.1 Introduction

The EFR32xG22 has several oscillators. This chapter contains a detailed function description and register descriptions for each oscillator. The CMU chapter includes information on how to select clock sources. Each oscillator may require some initial configuration or calibration before being enabled. The CMU supports clock on demand and can enable and disable oscillators. Therefore, it is important to properly configure each oscillator before selecting it as a clock source in the CMU.

9.2 HFXO - High Frequency Crystal Oscillator

9.2.1 Introduction

The High Frequency Crystal Oscillator (HFXO) uses an external high frequency crystal and provides a sequencer for starting up the crystal safely and reliably, while minimize energy consumption. An external sine wave clock source can also be used in the absence of a crystal.

9.2.2 Features

- Optimized for 38.4 MHz crystals
- Multiple programming options of start-up parameters to enable optimization of different crystals, supporting a wide range of ESR and ESL
- Programmable two-phase start-up to minimize energy consumption
- Built-in current optimization (Automatic oscillation amplitude control)
- Independent on-chip frequency tuning capacitors
- Hardware request for on-demand enable/disable
- Register lock

9.2.3 Functional Description

9.2.3.1 Enabling and Disabling

While the HFXO supports on-demand clocking, it is generally recommended to manually manage the HFXO, at least initially, because it requires software configuration and has a long start-up time. Software can set the FORCEEN to start HFXO and keep it enabled even if it is not selected as a clock source.

However, once started and before EM2 entry, switching the HFXO to on-demand mode may be desirable. This allows the MCU to enter EM2 and then restart the HFXO automatically upon EM2 exit. (During EM1P the HFXO can be conditionally started, depending on the wake-up trigger source.)

The HFXO can be enabled and disabled via both hardware and software mechanisms. Enabling via software is done by setting the FORCEEN bit in the HFXO_CTRL register. Disabling via software is done by setting the DISONDEMAND bit and clearing FORCEEN bit in the HFXO_CTRL register. The hardware controlled on-demand mode is enabled by clearing the FORCEEN and DISONDEMAND bits in the HFXO_CTRL register. Once configured the on-demand mode hardware can autonomously start and stop the HFXO based on various peripheral clock requests in combination with clock switch selections in the CMU. The HFXO is automatically stopped when entering EM2, EM3, or EM4. Hardware can also stop the HFXO via hardware in response to change in peripheral requests and clock switch selections in the CMU.

9.2.3.2 Start-up Time

The start-up time differs for different crystals and the HFXO has a configurable time-out to accommodate each crystal type. Software configures the timeout by setting the various TIMEOUT bit fields of the HFXO_XTALCFG register. The time-out delays the assertion of the RDY signal for HFXO. The programmed timeout should allow enough time for the oscillator to stabilize. The time-out can be optimized for the chosen crystal used in the application.

The start-up behavior of the HFXO also depends on how and how long the HFXO is disabled.

9.2.3.3 Configuration

The High Frequency Crystal Oscillator needs to be configured to ensure safe start-up for the given crystal. Refer to the Device Data sheet and application notes for guidelines in selecting correct components and crystals as well as for configuration trade-offs.

The HFXO crystal is connected to the HFXTAL_I/HFXTAL_O pins as shown in Figure 9.1 HFXO Pin Connection on page 173.

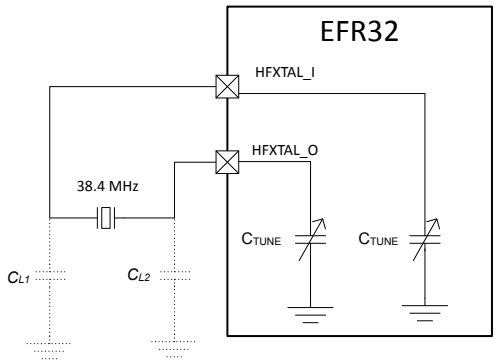


Figure 9.1. HFXO Pin Connection

Upon enabling the HFXO, a hardware state machine sequentially applies the configurable start-up state, intermediate start-up state, and steady state control settings from the HFXO_XTALCFG and HFXO_XTALCTRL registers. After reaching the steady operation state of the HFXO, it is recommended to further optimize current consumption using the Core Bias Optimization Algorithm to trade off noise and current consumption.

Refer to AN0016.2 for more information on settings for different crystals. Write the configuration values, which depends on the crystal's CL, RESR and oscillation frequency, into HFXO_XTALCFG and HFXO_XTALCTRL registers.

- COREBIASSTARTUP (HFXO_XTALCFG) - current setting applied at start-up time
- COREBIASSTARTUPI (HFXO_XTALCFG) - current setting applied at intermediate start-up time
- COREBIASANA (HFXO_XTALCTRL) - current setting applied at steady state
- CTUNEXISTARTUP (HFXO_XTALCFG) - tuning cap setting for XI applied at start-up time
- CTUNEXIANA (HFXO_XTALCTRL) - tuning cap setting for XI applied at steady state
- CTUNEXOSTARTUP (HFXO_XTALCFG) - tuning cap setting for XO applied at start-up time
- CTUNEXOANA (HFXO_XTALCTRL) - tuning cap setting for XO applied at steady state
- CTUNEFIXANA (HFXO_XTALCTRL) - fixed tuning cap setting applied throughout
- TIMEOUTSTEADY (HFXO_XTALCFG) - duration for the steady state settling time
- TIMEOUTCBLSB (HFXO_XTALCFG) - duration for the optimization settling after each step

All HFXO configuration needs to be performed prior to enabling the HFXO, whether via software by setting FORCEEN bit field, or allowing hardware request by clearing DISONDEMAND bit field in the HFXO_CTRL register.

By default, the HFXO is started in crystal mode, but it is possible to connect an active external sine or clipped sine wave clock source to the HFXTAL_I pin of the HFXO. By configuring the MODE field in HFXO_CFG to EXTCLK, the HFXO can be bypassed and the source clock can be provided through the HFXTAL_I pin.

9.2.3.4 Status Flags

The ENS flag in the HFXO_STATUS indicates if the HFXO has been successfully enabled. Once the HFXO oscillation amplitude has exceeded the start-up threshold and intermediate start-up threshold, the steady state settling timeout begins. When the steady state timeout has expired, the HFXO is ready for use as indicated by the RDY flag in the HFXO_STATUS. Once Core Bias Optimization is enabled, the COREBIASOPTRDY flag in the CMU_STATUS register indicates when the optimization is ready. It is advised to wait for this flag before using the HFXO, because optimization can cause minor disturbance to the oscillator frequency.

9.2.3.5 On-Demand Clocking

Hardware can request to enable the HFXO by setting the HFXO_STATUS.HWREQ bit field. The HFXO can also optionally be configured via the HFXO_STATUS.DISONDEMAND to shut down when no hardware request is present. This is known as on-demand clocking and allows the oscillator to be controlled without any software intervention. On-demand HFXO enable can be used, for example, upon wake-up of the Radio Controller (RAC). The RAC module always requires the HFXO for its operation. Any hardware request for HFXO, including request from RAC, is indicated in the HWREQ bit field of the HFXO_STATUS register. This request enables the HFXO, provided that DISONDEMAND bit field is cleared in HFXO_CTRL register. The HFXO is only disabled by hardware upon EM2, EM3 or EM4 entry.

A typical use scenario of the on-demand feature is as follows. Set up the PRORTC to periodically generate a compare match. Setup a PRS channel which uses this PRORTC compare match as its source to cause a wake-up into EM1. Setup the RAC to use the PRS channel as its source for TXEN or RXEN. Now, when the EFR32 is in EM2 and the RTCC generates a compare match, a wake-up into EM1 occurs, and the HFXO will automatically start. When HFXO is ready, the RAC performs its work and triggers a transition back into EM2 when finished. The system starts, uses, and stops the HFXO without ever being in EM0.

The HFXO analog circuitry can optionally continue operating with the clock output shut off when the HFXO is disabled. This is configured by setting the KEEPWARM bit in HFXO_STATUS.

9.2.3.6 Interrupts

RDYIF and COREBIASOPTRDYIF are interrupt flags as well as status flags. This allows software flexibility to implement interrupt service routine or polling loop for these events. When steady state timeout has exceeded, sticky RDYIF is set until it is cleared by software. If optimization is enabled, sticky COREBIASOPTRDYIF is set when optimization is completed successfully. However, if optimization fails to complete, sticky COREBIASOPTERRIF is set, and the HFXO control state machine stays in the error state until the oscillator is disabled. Similarly, if HFXO fails to start-up, meaning it has not reached the steady state, sticky DNSERRIF is set. The HFXO control state machine stays in the error state until the oscillator is disabled.

9.2.3.7 Protection

It is possible to lock the control registers, configuration registers, and command register to prevent unintended software writes to critical clock settings. This is controlled by the HFXO_LOCK register. A LOCK bit is available in HFXO_STATUS register. Furthermore, these registers are locked automatically by hardware to prevent clock domain crossing malfunction. To gain access to these registers while oscillator is in steady operation state, set FORCEEN to 1, then set DISONDEMAND to 1 in the HFXO_CTRL register. Next, issue the MANUALOVERRIDE command in the HFXO_CMD register to update the control registers and configuration registers with FSM values, allowing software direct control of the oscillator. A FSMLOCK bit in HFXO_STATUS register indicates when it is safe for software to update control registers and configuration registers. When software is finished with updates, put the oscillator back to on-demand mode by clearing DISONDEMAND to 0, followed by clearing FORCEEN to 0 in the HFXO_CTRL register. While DISONDEMAND is 0, FSMLOCK is always set, even if hardware is not requesting. This is to prevent a race condition between software access and hardware lock.

9.2.3.8 Tuning

While the oscillator is running in steady operation state, it may be desirable to change control settings. One example is frequency tuning by modifying the tuning capacitance via CTUNEXIANA and CTUNEXOANA fields in the HFXO_XTALCTRL register. When tuning, care should be taken to make small changes to the CTUNE registers. Ideally, change the CTUNE registers by one LSB at a time and alternate between the XI and XO registers. Sufficient wait time for settling, on the order of TIMEOUTSTEADY, should pass before new frequency measurement is taken.

Note: While the HFXO can support crystals with a tuning range of 38 MHz to 40 MHz, the radio specifically requires a 38.4 MHz MHz crystal. There may also be specific crystal tolerance requirements for each RF protocol supported by the radio.

9.2.4 HFXO Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	HFXO_IPVERSION	R	IP version ID
0x010	HFXO_XTALCFG	RWH	Crystal Configuration Register
0x018	HFXO_XTALCTRL	RWH	Crystal Control Register
0x020	HFXO_CFG	RWH	Configuration Register
0x028	HFXO_CTRL	RWH	Control Register
0x050	HFXO_CMD	W	Command Register
0x058	HFXO_STATUS	RH	Status Register
0x070	HFXO_IF	RWH INTFLAG	Interrupt Flag Register
0x074	HFXO_IEN	RW	Interrupt Enable Register
0x080	HFXO_LOCK	W	Configuration Lock Register
0x1000	HFXO_IPVERSION_SET	R	IP version ID
0x1010	HFXO_XTALCFG_SET	RWH	Crystal Configuration Register
0x1018	HFXO_XTALCTRL_SET	RWH	Crystal Control Register
0x1020	HFXO_CFG_SET	RWH	Configuration Register
0x1028	HFXO_CTRL_SET	RWH	Control Register
0x1050	HFXO_CMD_SET	W	Command Register
0x1058	HFXO_STATUS_SET	RH	Status Register
0x1070	HFXO_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1074	HFXO_IEN_SET	RW	Interrupt Enable Register
0x1080	HFXO_LOCK_SET	W	Configuration Lock Register
0x2000	HFXO_IPVERSION_CLR	R	IP version ID
0x2010	HFXO_XTALCFG_CLR	RWH	Crystal Configuration Register
0x2018	HFXO_XTALCTRL_CLR	RWH	Crystal Control Register
0x2020	HFXO_CFG_CLR	RWH	Configuration Register
0x2028	HFXO_CTRL_CLR	RWH	Control Register
0x2050	HFXO_CMD_CLR	W	Command Register
0x2058	HFXO_STATUS_CLR	RH	Status Register
0x2070	HFXO_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2074	HFXO_IEN_CLR	RW	Interrupt Enable Register
0x2080	HFXO_LOCK_CLR	W	Configuration Lock Register
0x3000	HFXO_IPVERSION_TGL	R	IP version ID
0x3010	HFXO_XTALCFG_TGL	RWH	Crystal Configuration Register
0x3018	HFXO_XTALCTRL_TGL	RWH	Crystal Control Register
0x3020	HFXO_CFG_TGL	RWH	Configuration Register
0x3028	HFXO_CTRL_TGL	RWH	Control Register

Offset	Name	Type	Description
0x3050	HFXO_CMD_TGL	W	Command Register
0x3058	HFXO_STATUS_TGL	RH	Status Register
0x3070	HFXO_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3074	HFXO_IEN_TGL	RW	Interrupt Enable Register
0x3080	HFXO_LOCK_TGL	W	Configuration Lock Register

9.2.5 HFXO Register Description

9.2.5.1 HFXO_IPVERSION - IP version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x2																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x2	R	**IP Version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

9.2.5.2 HFXO_XTALCFG - Crystal Configuration Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x4				0x4				0x3				0x3				0x13						0xB					
Access					RW				RW				RW				RW				RW						RW					
Name					TIMEOUTCBLSB				TIMEOUTSTEADY				CTUNEXOSTARTUP				CTUNEXISTARTUP				COREBIASSTARTUP						COREBIASSTARTUPI					

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:24	TIMEOUTCBLSB	0x4	RW	**Core Bias LSB Change Timeout**

wait duration for the COREBIAS change to settle out, used at each step of COREBIAS optimization algorithm

Value	Mode	Description
0	T8US	The core bias LSB change timeout is set to 8 us minimum. The maximum can be +40%.
1	T20US	The core bias LSB change timeout is set to 20 us minimum. The maximum can be +40%.
2	T41US	The core bias LSB change timeout is set to 41 us minimum. The maximum can be +40%.
3	T62US	The core bias LSB change timeout is set to 62 us minimum. The maximum can be +40%.
4	T83US	The core bias LSB change timeout is set to 83 us minimum. The maximum can be +40%.
5	T104US	The core bias LSB change timeout is set to 104 us minimum. The maximum can be +40%.
6	T125US	The core bias LSB change timeout is set to 125 us minimum. The maximum can be +40%.
7	T166US	The core bias LSB change timeout is set to 166 us minimum. The maximum can be +40%.
8	T208US	The core bias LSB change timeout is set to 208 us minimum. The maximum can be +40%.
9	T250US	The core bias LSB change timeout is set to 250 us minimum. The maximum can be +40%.
10	T333US	The core bias LSB change timeout is set to 333 us minimum. The maximum can be +40%.
11	T416US	The core bias LSB change timeout is set to 416 us minimum. The maximum can be +40%.
12	T833US	The core bias LSB change timeout is set to 833 us minimum. The maximum can be +40%.

Bit	Name	Reset	Access	Description
	13	T1250US		The core bias LSB change timeout is set to 1250 us minimum. The maximum can be +40%.
	14	T2083US		The core bias LSB change timeout is set to 2083 us minimum. The maximum can be +40%.
	15	T3750US		The core bias LSB change timeout is set to 3750 us minimum. The maximum can be +40%.
23:20	TIMEOUTSTEADY	0x4	RW	**Steady State Timeout**
	wait duration for the steady state settings to settle out			
	Value	Mode		Description
	0	T16US		The steady state timeout is set to 16 us minimum. The maximum can be +40%.
	1	T41US		The steady state timeout is set to 41 us minimum. The maximum can be +40%.
	2	T83US		The steady state timeout is set to 83 us minimum. The maximum can be +40%.
	3	T125US		The steady state timeout is set to 125 us minimum. The maximum can be +40%.
	4	T166US		The steady state timeout is set to 166 us minimum. The maximum can be +40%.
	5	T208US		The steady state timeout is set to 208 us minimum. The maximum can be +40%.
	6	T250US		The steady state timeout is set to 250 us minimum. The maximum can be +40%.
	7	T333US		The steady state timeout is set to 333 us minimum. The maximum can be +40%.
	8	T416US		The steady state timeout is set to 416 us minimum. The maximum can be +40%.
	9	T500US		The steady state timeout is set to 500 us minimum. The maximum can be +40%.
	10	T666US		The steady state timeout is set to 666 us minimum. The maximum can be +40%.
	11	T833US		The steady state timeout is set to 833 us minimum. The maximum can be +40%.
	12	T1666US		The steady state timeout is set to 1666 us minimum. The maximum can be +40%.
	13	T2500US		The steady state timeout is set to 2500 us minimum. The maximum can be +40%.
	14	T4166US		The steady state timeout is set to 4166 us minimum. The maximum can be +40%.
	15	T7500US		The steady state timeout is set to 7500 us minimum. The maximum can be +40%.
19:16	CTUNEXOSTARTUP	0x3	RW	**Startup Tuning Capacitance on XO**
	4 most significant bits of CTUNEXOANA applied during startup phase			
15:12	CTUNEXISTARTUP	0x3	RW	**Startup Tuning Capacitance on XI**

Bit	Name	Reset	Access	Description
	4 most significant bits of CTUNEXIANA applied during startup phase			
11:6	COREBIASSTARTUP	0x13	RW	**Startup Core Bias Current**
	6 most significant bits of COREBIASANA applied during startup phase			
5:0	COREBIASSTARTUPI	0xB	RW	**Intermediate Startup Core Bias Current**
	6 most significant bits of COREBIASANA applied during intermediate startup phase			

9.2.5.3 HFXO_XTALCTRL - Crystal Control Register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0				0x3		0x3		0x8C								0x8C								0x10							
Access	RW				RW		RW		RW								RW								RW							
Name	SKIPCOREBIASOPT				COREDGENANA		CTUNEFIXANA		CTUNEXOANA								CTUNEXIANA								COREBIASANA							

Bit	Name	Reset	Access	Description
31	SKIPCOREBIASOPT	0x0	RW	**Skip Core Bias Optimization**
	Set to skip the core bias current optimization algorithm at next startup. Reuse the value stored in COREBIASANA. At the successful completion of core bias current optimization algorithm, hardware sets this bit to skip optimization during subsequent startup.			
30:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:26	COREDGENANA	0x3	RW	**Core Degeneration**
	Core degeneration control			

Value	Mode	Description
0	NONE	Do not apply core degeneration resistence
1	DGEN33	Apply 33 ohm core degeneration resistence
2	DGEN50	Apply 50 ohm core degeneration resistence
3	DGEN100	Apply 100 ohm core degeneration resistence

Bit	Name	Reset	Access	Description
25:24	CTUNEFIXANA	0x3	RW	**Fixed Tuning Capacitance**
	Adds or removes fixed capacitance on XI or XO			

Value	Mode	Description
0	NONE	Remove fixed capacitance on XI and XO nodes
1	XI	Adds fixed capacitance on XI node
2	XO	Adds fixed capacitance on XO node
3	BOTH	Adds fixed capacitance on both XI and XO nodes

Bit	Name	Reset	Access	Description
23:16	CTUNEXOANA	0x8C	RW	**Tuning Capacitance on XO**
	Approximately 80fF per step. 0 is min. 255 is max.			
15:8	CTUNEXIANA	0x8C	RW	**Tuning Capacitance on XI**
	Approximately 80fF per step. 0 is min. 255 is max.			
7:0	COREBIASANA	0x10	RW	**Core Bias Current**

Bit	Name	Reset	Access	Description
	Approximately 10uA per step			

9.2.5.4 HFXO_CFG - Configuration Register

Offset	Bit Position																														
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0		0x0
Access																													RW	RW		RW
Name																													SQBUFSCHTRGANA	ENXIDCBIASANA		MODE

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	SQBUFSCHTRGANA	0x0	RW	**Squaring Buffer Schmitt Trigger**
	Used in EXTCLK mode to prevent self oscillation			

Value	Mode	Description
0	DISABLE	Squaring buffer schmitt trigger is disabled
1	ENABLE	Squaring buffer schmitt trigger is enabled

Bit	Name	Reset	Access	Description
2	ENXIDCBIASANA	0x0	RW	**Enable XI Internal DC Bias**
	Set to enable internal DC bias. Bit is ignored in XTAL mode.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	MODE	0x0	RW	**Crystal Oscillator Mode**
	Set this to configure the external source for the HFXO.			

Value	Mode	Description
0	XTAL	crystal oscillator
1	EXTCLK	external sinusoidal clock can be supplied on XI pin.

9.2.5.5 HFXO_CTRL - Control Register

Offset	Bit Position																															
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0		0x0	0x1	0x0
Access																											RW	RW		RW	RW	RW
Name																											FORCEXO2GNDANA	FORCEXI2GNDANA		KEEPWARM	DISONDEMAND	FORCEEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	FORCEXO2GNDANA	0x0	RW	**Force XO Pin to Ground**
	Set to enable grounding of XO pin.			
	Value	Mode		Description
	0	DISABLE		Disabled (not pulled)
	1	ENABLE		Enabled (pulled)
4	FORCEXI2GNDANA	0x0	RW	**Force XI Pin to Ground**
	Set to enable grounding of XI pin. Do not enable if MODE=EXTCLK and an external source is supplied.			
	Value	Mode		Description
	0	DISABLE		Disabled (not pulled)
	1	ENABLE		Enabled (pulled)
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	KEEPWARM	0x0	RW	**Keep Warm**
	Upon disable, if this bit is set, analog oscillator will keep running, while clock output is shutoff. Clearing this bit has no effect until the next disable event.			
1	DISONDEMAND	0x1	RW	**Disable On-demand Mode**
	Set to ignore hardware request enabling the crystal oscillator. This bit must be set in order to modify various CFG and CTRL registers while FSMLOCK is deasserted.			
0	FORCEEN	0x0	RW	**Force Enable**
	Force the oscillator to run even without a hardware request.			

9.2.5.6 HFXO_CMD - Command Register

Offset	Bit Position																													
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															W	W
Name																															MANUALOVERRIDE	COREBIASOPT

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	MANUALOVERRIDE	0x0	W	**Manual Override**
	Updates APB registers with FSM values. When update completes, switch analog control signals from FSM to APB, allowing software to control the oscillator directly. Only issue this command when COREBIASOPTRDY is asserted, or the command is ignored.			
0	COREBIASOPT	0x0	W	**Core Bias Optimizaton**
	Starts the core bias current optimization algorithm and runs it one time. Optimization should be executed if the temperature changes by more than 40degC. Do not run this command while the radio is in RX or TX modes. Do not issue this command more than once until COREBIASOPTRDY is asserted, or the previous command may be cancelled.			

9.2.5.7 HFXO_STATUS - Status Register

Offset	Bit Position																													
0x058	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0											0x0		0x0	0x0															0x0	0x0
Access	R	R											R		R	R															R	R
Name	LOCK	FSMLOCK											ISWARM		HWREQ	ENS															COREBIASOPTRDY	RDY

Bit	Name	Reset	Access	Description
31	LOCK	0x0	R	**Configuration Lock Status**
	Indicates the current status of configuration lock			
	Value	Mode		Description
	0	UNLOCKED		Configuration lock is unlocked
	1	LOCKED		Configuration lock is locked
30	FSMLOCK	0x0	R	**FSM Lock Status**
	Indicates the current status of configuration locked by FSM running			
	Value	Mode		Description
	0	UNLOCKED		FSM lock is unlocked
	1	LOCKED		FSM lock is locked
29:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	ISWARM	0x0	R	**Oscillator Is Kept Warm**
	Oscillator is currently kept in warm state. Re-eable from warm state skips startup sequence			
18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	HWREQ	0x0	R	**Oscillator Requested by Hardware**
	Oscillator is requested by hardware.			
16	ENS	0x0	R	**Enabled Status**
	Oscillator is enabled.			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	COREBIASOPTRDY	0x0	R	**Core Bias Optimization Ready**
	Core bias optimization algorithm is complete. New core bias value updated to XTALCTRL.			
0	RDY	0x0	R	**Ready Status**
	Oscillator is enabled and start-up time has exceeded.			

9.2.5.8 HFXO_IF - Interrupt Flag Register

Offset	Bit Position																													
0x070	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0		0x0																												0x0	0x0
Access	RW		RW																												RW	RW
Name	COREBIASOPTERR		DNSERR																												COREBIASOPTRDY	RDY

Bit	Name	Reset	Access	Description
31	COREBIASOPTERR	0x0	RW	**Core Bias Optimization Error Interrupt**
	Core bias current optimization algorithm fails to complete.			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	DNSERR	0x0	RW	**Did Not Start Error Interrupt**
	Crystal oscillator fails to startup.			
28:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	COREBIASOPTRDY	0x0	RW	**Core Bias Optimization Ready Interrupt**
	Core bias current optimization algorithm is complete.			
0	RDY	0x0	RW	**Ready Interrupt**
	Oscillator is ready (start-up time exceeded).			

9.2.5.9 HFXO_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x074	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0		0x0																												0x0	0x0
Access	RW		RW																												RW	RW
Name	COREBIASOPTERR		DNSERR																												COREBIASOPTRDY	RDY

Bit	Name	Reset	Access	Description
31	COREBIASOPTERR	0x0	RW	**Core Bias Optimization Error Interrupt**
	Core bias current optimization algorithm fails to complete.			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	DNSERR	0x0	RW	**Did Not Start Error Interrupt**
	Crystal oscillator fails to startup.			
28:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	COREBIASOPTRDY	0x0	RW	**Core Bias Optimization Ready Interrupt**
	Core bias current optimization algorithm is complete.			
0	RDY	0x0	RW	**Ready Interrupt**
	Oscillator is ready (start-up time exceeded).			

9.2.5.10 HFXO_LOCK - Configuration Lock Register

Offset	Bit Position																													
0x080	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x580E						
Access																									W						
Name																									LOCKKEY						

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x580E	W	**Configuration Lock Key** Write any other value than the unlock code to lock registers from editing. Write the unlock code to unlock.

Value	Mode	Description
22542	UNLOCK	Write this value to unlock

9.3 HFRCO - High-Frequency RC Oscillator

9.3.1 Introduction

The HFRCO is a calibrated internal High Frequency RC oscillator.

9.3.2 Features

- 1 MHz - 80 MHz High Frequency RC Oscillator with DPLL working in EM01 (HFRCODPLL)
- Low start-up time
- Run-time band change or tuning

9.3.3 Functional Description

9.3.3.1 Start-up

The HFRCO starts up quickly in a few micro-seconds (refer to device data sheet for start-up time specifications.) After the start-up time, the RDY status bit will go high and the RDY interrupt will be triggered. It can take another two clock cycles for the clock to propagate through the CMU before the clock is seen by peripherals.

9.3.3.2 On-Demand Clocking

Hardware can request to enable the HFRCO by setting the HFRCO_STATUS.HWREQ bit field. The HFRCO can also optionally be configured via the HFRCO_STATUS.DISONDEMAND to shut down when no hardware request is present. This is known as on-demand clocking and allows the oscillator to be controlled without any software intervention. This means that HFRCO receives a request for clock from the CMU whenever the oscillator clock is needed. These requests can come at any time from any power domain (depending on the which peripheral is requesting the clock.)

9.3.3.3 Calibration

Several different frequencies are calibrated during production test on every device. In order to use a factory-calibrated value, software must read the value from the appropriate location in the DEVINFO page and write it to the CAL register.

The TUNING and FINETUNING bit fields in the CAL register can be used to trim HFRCO manually.

Software may write the CAL register at any time. If there is already a frequency updating occuring, the current change would apply when the previous update is done. FREQBSY in STATUS register indicates if the updating is finished.

The minimum and maximum frequencies attainable for each setting of the FREQRANGE field are listed in the device data sheet.

Table 9.1. HFRCODPLL Calibration Frequencies

DEVINFO Location	Target Frequency
HFRCODPLLCAL0	4 MHz
HFRCODPLLCAL1	5 MHz
HFRCODPLLCAL3	7 MHz
HFRCODPLLCAL4	10 MHz
HFRCODPLLCAL6	13 MHz
HFRCODPLLCAL7	16 MHz
HFRCODPLLCAL8	19 MHz (default)
HFRCODPLLCAL9	20 MHz
HFRCODPLLCAL10	26 MHz
HFRCODPLLCAL11	32 MHz
HFRCODPLLCAL12	38 MHz
HFRCODPLLCAL13	48 MHz
HFRCODPLLCAL14	56 MHz
HFRCODPLLCAL15	64 MHz
HFRCODPLLCAL16	80 MHz

9.3.3.4 Interrupts

HFRCO has one interrupt: IF.RDY. RDY is triggered when the timeout has finished and the qualified HFRCO clock is ready. The clock is gated until it is ready.

9.3.3.5 Status Flags

9.3.3.5.1 FREQBSY

The FREQBSY bit indicates the HFRCO is busy updating its frequency after writing to the CAL register. The FREQBSY bit should be used whenever frequency is changed. E.g. After software writes to the CAL register, FREQBSY would assert immediately. Software should wait for FREQBSY to be zero before attempting to write to the CAL register again.

For band-change, FREQBSY would not de-assert until after the timeout upon being re-enabled.

For normal start-up, FREQBSY would not assert.

When DPLL is on, FREQBSY would not assert as the frequency change is not caused by writing to the CAL register. When disabling DPLL the last tuning value is written back to the CAL register, which will assert FREQBSY.

9.3.3.5.2 ENS

ENS indicates the HFRCO is enabled. This flag is used to check if the HFRCO is enabled by any requester.

Note: When a band change occurs, the HFRCO is disabled and re-enabled. This will cause the ENS bit to briefly de-assert.

9.3.3.5.3 RDY

RDY indicates HFRCO is enabled and start-up timeout has exceeded. Used to check if the HFRCO clock is ready after enable.

Band-change would de-assert RDY as it would go through another

9.3.3.5.4 SYNCBUSY

SYNCBUSY indicates ongoing synchronization of CAL register fields. Same as all other modules.

9.3.3.6 Forced Oscillator Control

The HFRCO can be forced on and off using the FORCEEN and DISONDEMAND bits in the CTRL register.

Setting FORCEEN will force the oscillator core to run, but peripherals will still need to request the clock to un-gate the clock signal.

9.3.3.7 Oscillator Modes

The HFRCO has three modes of operation, an **on-demand** mode (which is the normal software use case), a **force on** and a **force off** mode.

In **on-demand** mode the oscillator will start whenever a peripheral requests the it. Which in most cases is whenever the module is enabled.

In **force on** mode the analog core will run independently of whether it is requested or not. This can be useful for measuring analog current without any digital load on the clocks.

In **force off** mode, the analog core will be shut off independently of whether it is requested or not. This can be useful for changing analog test settings without risking glitches on the clock.

The DISONDEMAND bit can also be used to give software full control over the clock for exceptional cases where software control is desired.

Table 9.2. Oscillator modes

Bit Field	FORCEEN	DISONDEMAND
On-Demand (normal operation)	0	0
Forced On	1	X
Forced Off	0	1

9.3.4 HFRCO Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	HFRCO_IPVERSION	R	IP Version ID
0x004	HFRCO_CTRL	RW	Ctrl Register
0x008	HFRCO_CAL	RWH SYNC	Calibration Register
0x00C	HFRCO_STATUS	RH	Status Register
0x010	HFRCO_IF	RWH INTFLAG	Interrupt Flag Register
0x014	HFRCO_IEN	RW	Interrupt Enable Register
0x01C	HFRCO_LOCK	W	Lock Register
0x1000	HFRCO_IPVERSION_SET	R	IP Version ID
0x1004	HFRCO_CTRL_SET	RW	Ctrl Register
0x1008	HFRCO_CAL_SET	RWH SYNC	Calibration Register
0x100C	HFRCO_STATUS_SET	RH	Status Register
0x1010	HFRCO_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1014	HFRCO_IEN_SET	RW	Interrupt Enable Register
0x101C	HFRCO_LOCK_SET	W	Lock Register
0x2000	HFRCO_IPVERSION_CLR	R	IP Version ID
0x2004	HFRCO_CTRL_CLR	RW	Ctrl Register
0x2008	HFRCO_CAL_CLR	RWH SYNC	Calibration Register
0x200C	HFRCO_STATUS_CLR	RH	Status Register
0x2010	HFRCO_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2014	HFRCO_IEN_CLR	RW	Interrupt Enable Register
0x201C	HFRCO_LOCK_CLR	W	Lock Register
0x3000	HFRCO_IPVERSION_TGL	R	IP Version ID
0x3004	HFRCO_CTRL_TGL	RW	Ctrl Register
0x3008	HFRCO_CAL_TGL	RWH SYNC	Calibration Register
0x300C	HFRCO_STATUS_TGL	RH	Status Register
0x3010	HFRCO_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3014	HFRCO_IEN_TGL	RW	Interrupt Enable Register
0x301C	HFRCO_LOCK_TGL	W	Lock Register

9.3.5 HFRCO Register Description

9.3.5.1 HFRCO_IPVERSION - IP Version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x1	R	**IP Version**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

9.3.5.2 HFRCO_CTRL - Ctrl Register

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															DISONDEMAND	FORCEEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	DISONDEMAND	0x0	RW	**Disable On-demand**
	Setting this bit disable HFRCO on-demand feature			
0	FORCEEN	0x0	RW	**Force Enable**
	Setting this bit force HFRCO enabled			

9.3.5.3 HFRCO_CAL - Calibration Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xA				0x2		0x0		0x3			0x8					0x1		0x1F							0x7F						
Access	RW				RW		RW		RW			RW					RW		RW							RW						
Name	IREFTC				CMPSEL		CLKDIV		CMPBIAS			FREQRANGE					LDOHP		FINETUNING							TUNING						

Bit	Name	Reset	Access	Description
31:28	IREFTC	0xA	RW	**Tempco Trim on Comparator Current**
	Writing this field adjusts the temperature coefficient trim on comparator current.			
27:26	CMPSEL	0x2	RW	**Comparator Load Select**
	Writing this field adjusts the active load for comparators.			
25:24	CLKDIV	0x0	RW	**Locally Divide HFRCO Clock Output**
	Writing this field configures the HFRCO clock output divider.			

Value	Mode	Description
0	DIV1	Divide by 1.
1	DIV2	Divide by 2.
2	DIV4	Divide by 4.

23:21	CMPBIAS	0x3	RW	**Comparator Bias Current**
	Writing this field adjusts the HFRCO comparator bias current.			
20:16	FREQRANGE	0x8	RW	**Frequency Range**
	Writing this field adjusts the HFRCO frequency range.			
15	LDOHP	0x1	RW	**LDO High Power Mode**
	Settings this bit puts the HFRCO LDO in high power mode.			
14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	FINETUNING	0x1F	RW	**Fine Tuning Value**
	Writing this field adjusts the HFRCO fine tuning value. Higher value means lower frequency.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	TUNING	0x7F	RW	**Tuning Value**
	Writing this field adjusts the HFRCO tuning value. Higher value means lower frequency.			

9.3.5.4 HFRCO_STATUS - Status Register

Offset	Bit Position																														
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0															0x0														0x0	0x0	0x0
Access	R															R														R	R	R
Name	LOCK															ENS														SYNCBUSY	FREQBSY	RDY

Bit	Name	Reset	Access	Description
31	LOCK	0x0	R	**Lock Status**
	If set, all HFRCO lockable registers are locked.			

Value	Mode	Description
0	UNLOCKED	HFRCO is unlocked
1	LOCKED	HFRCO is locked

30:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	ENS	0x0	R	**Enable Status**
	HFRCO is enabled.			
15:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	SYNCBUSY	0x0	R	**Synchronization Busy**
	This bit is set when there is an ongoing synchronization of CAL register bitfields.			
1	FREQBSY	0x0	R	**Frequency Updating Busy**
	HFRCO is busy updating frequency.			
0	RDY	0x0	R	**Ready**
	HFRCO is enabled and start-up time has exceeded.			

9.3.5.5 HFRCO_IF - Interrupt Flag Register

Offset	Bit Position																														
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																RDY

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	RDY	0x0	RW	**Ready Interrupt Flag**
	Set when HFRCO is ready (start-up time exceeded).			

9.3.5.6 HFRCO_IEN - Interrupt Enable Register

Offset	Bit Position																														
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																RDY

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	RDY	0x0	RW	**RDY Interrupt Enable**
	Enable/disable the RDY interrupt			

9.3.5.7 HFRCO_LOCK - Lock Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x8195						
Access																									W						
Name																									LOCKKEY						

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x8195	W	**Lock Key**

Write any other value than the unlock code to lock registers from editing. Write the unlock code to unlock.

Value	Mode	Description
33173	UNLOCK	Unlock code

9.4 DPLL - Digital Phased Locked Loop

9.4.1 Introduction

The Digital Phase-Locked Loop (DPLL) uses a reference clock to generate a desired clock frequency at a specified ratio to the reference clock.

9.4.2 Features

- Frequency Lock Mode
- Phase-Lock Mode
- Output frequency = F_{REF}*(N+1)/(M+1), where N and M are 12-bit values
- Very fast lock time
- Very fast transient tracking
- Low output jitter
- Lock detection with an interrupt
- Lock fail detection with interrupts

9.4.3 Functional Description

9.4.3.1 Enabling and Disabling

The DPLL can be enabled and disabled by software via the DPLL_EN register. Before enabling DPLL, software should:

1. Select reference clock by setting the CLKSEL field in CMU_DPLLREFCLKCTRL.
2. The CMU should not be running from the HFRCO. If necessary, the CMU should switch to the FSRCO until after the DPLL has locked to avoid over-clocking due to overshoot. If necessary, select FRSCO or HFXO in the CMU_SYSCLKCTRL register CLKSEL field.
3. Configure the DPLL.
4. Make certain that the ENS bit in DPLL_STATUS is low.

The DPLL is disabled automatically when entering EM2, EM3, or EM4. Note that disabling the DPLL will not automatically turn off the reference clock. The CLKSEF field in CMU_DPLLREFCLKCTRL must be set to DISABLED before entering EM2 or the selected REFCLK may continue to run in EM2.

9.4.3.2 Lock Modes

The DPLL provides two lock modes, referred to as frequency-lock loop mode (FREQLL) and phase-lock loop mode (PHASELL). FREQLL mode keeps the DCO frequency-locked to the reference clock, which means the DCO frequency will be accurate. However, the phase error can accumulate over time and cause a non-zero average frequency error. FREQLL mode also provides better jitter and transient performance. PHASELL mode keeps the DCO phase-locked to the reference clock, which means the phase error does not accumulate over time, which makes the average frequency error zero. FREQLL mode should be used unless specific phase requirement exists.

9.4.3.3 Configurations

The formula for the DPLL output frequqncy is FREF*(N+1)/(M+1). The user should calculate N and M in DPLL_CFG1 to achieve the target frequency. Note that with a larger value of N, the DCO lock time would increase and DCO jitter would decrease. Both effects are approximately linear. This relationship can be used to select N for a given application to strike a compromise between lock time and output jitter. For example if an ratio of 3 is desired, the DPLL could be configured as {N=599, M=199} for fast lock time but high jitter, or as {N=2999, M=999} for lower jitter but longer lock time.

Note: All configuration settings should be done before enabling the DPLL. They should not be changed when DPLL is running. The final tuning values can be read back from TUNING and FINETUNING in HFRCO_CAL, after DPLL is disabled and DPLLENS in DPLL_STATUS is low.

9.4.3.4 Lock Detection

The DPLL has 3 different types of output events: ready, lock fail due to period underflow, and lock fail due to period overflow. Each of the events has its own interrupt flag. DPLLRDY is set when DPLL successfully locks to the reference clock based on the software configuration. DPLLLOCKFAILLOW is set when the DPLL fails to lock because the period lower boundary is hit. DPLLLOCKFAILHIGH is set when the DPLL fails to lock because the period upper boundary is hit. If the interrupt flags are set and the corresponding interrupt enable bits in DPLL_IEN are set, the DPLL will request an interrupt. Based on different interrupt events, software should take different actions:

- If the DPLLRDY interrupt is received first, it means target clock is ready and it is safe to switch to use DCO's output.
- If the DPLLLOCKFAILLOW interrupt is received first, it indicates the RANGE in HFRCO_CAL is too small. Software should disable the DPLL and write a larger value to RANGE, then enable the DPLL again to lock.
- If the DPLLLOCKFAILHIGH interrupt is received first, it indicates the RANGE in HFRCO_CAL is too large. Software should disable DPLL and write a smaller value to RANGE, then enable DPLL again to lock.
- If the DPLLRDY interrupt is received first and then DPLLLOCKFAILLOW or DPLLLOCKFAILHIGH is received later, it means reference clock drifted over 1% and the DPLL has lost its locked status.
  - If AUTORECOVER in DPLL_CFG is not set, software should disable the DPLL and enable DPLL again to lock.
  - If AUTORECOVER in DPLL_CFG is set, hardware will re-lock automatically. When the target frequency is near the boundary of a range, the drift may cause underflow or overflow. In this case the fail interrupt will still be received. Software should disable the DPLL and modify RANGE in HFRCO_CAL in corresponding direction, depending on whether the DPLLLOCKFAILLOW or DPLLLOCKFAILHIGH bit is set. Then enable DPLL again to lock.

9.4.4 DPLL Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	DPLL_IPVERSION	R	IP Version
0x004	DPLL_EN	RW ENABLE	Enable
0x008	DPLL_CFG	RW CONFIG	Config
0x00C	DPLL_CFG1	RW CONFIG	Config1
0x010	DPLL_IF	RWH INTFLAG	Interrupt Flag
0x014	DPLL_IEN	RW	Interrupt Enable
0x018	DPLL_STATUS	RH	Status
0x024	DPLL_LOCK	W	Lock
0x1000	DPLL_IPVERSION_SET	R	IP Version
0x1004	DPLL_EN_SET	RW ENABLE	Enable
0x1008	DPLL_CFG_SET	RW CONFIG	Config
0x100C	DPLL_CFG1_SET	RW CONFIG	Config1
0x1010	DPLL_IF_SET	RWH INTFLAG	Interrupt Flag
0x1014	DPLL_IEN_SET	RW	Interrupt Enable
0x1018	DPLL_STATUS_SET	RH	Status
0x1024	DPLL_LOCK_SET	W	Lock
0x2000	DPLL_IPVERSION_CLR	R	IP Version
0x2004	DPLL_EN_CLR	RW ENABLE	Enable
0x2008	DPLL_CFG_CLR	RW CONFIG	Config
0x200C	DPLL_CFG1_CLR	RW CONFIG	Config1
0x2010	DPLL_IF_CLR	RWH INTFLAG	Interrupt Flag
0x2014	DPLL_IEN_CLR	RW	Interrupt Enable
0x2018	DPLL_STATUS_CLR	RH	Status
0x2024	DPLL_LOCK_CLR	W	Lock
0x3000	DPLL_IPVERSION_TGL	R	IP Version
0x3004	DPLL_EN_TGL	RW ENABLE	Enable
0x3008	DPLL_CFG_TGL	RW CONFIG	Config
0x300C	DPLL_CFG1_TGL	RW CONFIG	Config1
0x3010	DPLL_IF_TGL	RWH INTFLAG	Interrupt Flag
0x3014	DPLL_IEN_TGL	RW	Interrupt Enable
0x3018	DPLL_STATUS_TGL	RH	Status
0x3024	DPLL_LOCK_TGL	W	Lock

9.4.5 DPLL Register Description

9.4.5.1 DPLL_IPVERSION - IP Version

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP Version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

9.4.5.2 DPLL_EN - Enable

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**Module Enable**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

9.4.5.3 DPLL_CFG - Config

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																										0x0				0x0	0x0	0x0
Access																										RW				RW	RW	RW
Name																										DITHEN				AUTORECOVER	EDGESEL	MODE

Bit	Name	Reset	Access	Description
31:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	DITHEN	0x0	RW	**Dither Enable Control**
	Set to enable dither function			
5:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	AUTORECOVER	0x0	RW	**Automatic Recovery Control**
	Set to enable automatic recovery function			
1	EDGESEL	0x0	RW	**Reference Edge Select**
	This bit controls which edge of reference is detected			
0	MODE	0x0	RW	**Operating Mode Control**
	This bit controls which mode DPLL is operating when enabled			

Value	Mode	Description
0	FLL	Frequency Lock Mode
1	PLL	Phase Lock Mode

9.4.5.4 DPLL_CFG1 - Config1

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0																0x0											
Access					RW																RW											
Name					N																M										

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:16	N	0x0	RW	**Factor N**
	The locked DCO frequency is given by: Fdco = Fref * (N + 1)/(M+1). N is required to be larger than 300.			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	M	0x0	RW	**Factor M**
	The locked DCO frequency is given by: Fdco = Fref * (N + 1)/(M+1). M can be any value.			

9.4.5.5 DPLL_IF - Interrupt Flag

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														RW	RW	RW
Name																														LOCKFAILHIGH	LOCKFAILLOW	LOCK

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	LOCKFAILHIGH	0x0	RW	**Lock Failure High Interrupt Flag**
	Set when DPLL fail to lock because of period overflow.			
1	LOCKFAILLOW	0x0	RW	**Lock Failure Low Interrupt Flag**
	Set when DPLL fail to lock because of period underflow.			
0	LOCK	0x0	RW	**Lock Interrupt Flag**
	Set when DPLL achieve the lock.			

9.4.5.6 DPLL_IEN - Interrupt Enable

Offset	Bit Position																														
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														RW	RW	RW
Name																														LOCKFAILHIGH	LOCKFAILLOW	LOCK

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	LOCKFAILHIGH	0x0	RW	**LOCKFAILHIGH Interrupt Enable**
	LOCKFAILHIGH Interrupt Enable			
1	LOCKFAILLOW	0x0	RW	**LOCKFAILLOW Interrupe Enable**
	LOCKFAILLOW Interrupe Enable			
0	LOCK	0x0	RW	**LOCK interrupt Enable**
	LOCK interrupt Enable			

9.4.5.7 DPLL_STATUS - Status

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																														0x0	0x0
Access	R																														R	R
Name	LOCK																														ENS	RDY

Bit	Name	Reset	Access	Description
31	LOCK	0x0	R	**Lock Status**
	Indicates the current status of configuration lock			

Value	Mode	Description
0	UNLOCKED	DPLL is unlocked
1	LOCKED	DPLL is locked

30:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	ENS	0x0	R	**Enable Status**
	DPLL is enabled.			
0	RDY	0x0	R	**Ready Status**
	DPLL is enabled and locked.			

9.4.5.8 DPLL_LOCK - Lock

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x7102															
Access																	W															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x7102	W	**Lock Key**
	Write any other value than the unlock code to lock registers from editing. Write the unlock code to unlock.			

Value	Mode	Description
28930	UNLOCK	Unlock code

9.5 LFXO - Low-Frequency Crystal Oscillator

9.5.1 Introduction

The Low Frequency Crystal Oscillator (LFXO) uses an external 32.768 kHz crystal to provide an accurate low-frequency clock. The module is available in all energy modes, except EM3. The main interaction is with the CMU through the clock requesting mechanism.

9.5.2 Features

High-level features.

- Crystal calibration
- Functional in all energy modes, except EM3
- Failure detection and EM4WU
- External CMOS mode
- Edge interrupts and EM2WU
- On-demand oscillator enabling

9.5.3 Functional Description

9.5.3.1 Modes

The LFXO can be used in three different modes. The mode can be programmed by setting MODE bit field in the LFXO_CFG register. If MODE is set to XTAL, the LFXO is programmed to operate in crystal mode and a 32.768 kHz crystal oscillator should be connected to LF crystal pads, LFXTAL_I and LFXTAL_O (see the device data sheet for details). If MODE is set to BUFEXTCLK, the LFXO is programmed to operate in external sine mode and the sine wave should be supplied to LFXTAL_I pin. If MODE is set to DIGEXTCLK, LFXO is programmed to operate in external CMOS mode and the external 32.768 kHz clock should be provided on LFXTAL_I pin. See the register descriptions for more details.

9.5.3.2 Enabling

There are two ways to turn on the LFXO clock. One is to turn it on in FORCEON mode by setting FORCEEN bit to 1 in LFXO_CTRL register. Another is to keep it ready to be turned on in ONDEMAND mode by setting FORCEEN bit to 0 and DISONDEMAND bit to 0 in LFXO_CTRL register. This means that the oscillator will be off unless its clock requested. When a peripheral requests the clock, hardware will automatically enable the LFXO without any software intervention. The oscillator will remain on as long as the peripheral requests it. DISONDEMAND setting does not have any impact when FORCEEN set to 1. LFXO is in FORCEOFF mode when FORCEEN set to 0 and DISONDEMAND set to 1. In FORCEOFF mode all requests are blocked and LFXO will not generate the clock. The LFXO clock is available in all energy modes, except EM3.

9.5.3.3 Clock Qualification

The clock should not be used immediately after enabling LFXO, until the clock has had time to stabilize. Therefore a number of cycles are required to qualify the clock. Before the clock is qualified, no clock requesters will receive the LFXO clock. The number of cycles used to qualify the clock can be programmed by setting TIMEOUT bit field in the LFXO_CFG register. The TIMEOUT default value is set the 32,728 cycles, which is much more than necessary for stabilization. The stabilization time required will depend on the particular crystal, oscillator settings, and frequency accuracy requirements. A value of 4096 clocks is generally recommended for most applications. A low timeout of 2 cycles may be used in DIGEXTCLK mode in order to filter out the first glitch from the pad. The 2 clock cycle timeout should not be used with crystals. There are two status bits and one interrupt associated with enabling the oscillator and qualifying its clock. Once the oscillator gets enabled the ENS bit in LFXO_STATUS register will be set high. Note that due to the nature of on demand clocking, the oscillator can be enabled anytime, so if software reads ENS low it is not safe to assume that ENS stays low during the next instruction. It is only safe to assume that oscillator is OFF at the time ENS is being read. Similarly, if software reads ENS high it is not safe to assume that ENS stays high during the next instruction. Once the clock is qualified, the RDY status is set high in the LFXO_STATUS register. The same uncertainties also apply to the RDY bit. However, software can wait for RDY bit to go high to detect that LFXO clock is qualified. Or it can enable the interrupt with RDYIEN in LFXO_IEN register and receive RDYIF interrupt available in LFXO_IF register. RDYIF also acts as EM2 wakeup source if RDYIEN set high. If put into FORCEON mode, the LFXO will start the qualification and once qualified it will gate off the clock but immediately start with no qualification upon receiving a request. If in ONDEMAND mode, the LFXO starts the qualification every time it is switched from off to on due to clock requests. The qualification can take up to 32k cycles. Note that only enabling RDY interrupt does not act as a clock request.

9.5.3.4 Edge Detection Interrupts

There is a possibility for software to detect rising or falling edges of the LFXO clock. The edge detection is enabled if any of POSEDGEIEN and NEGEDGEIEN is set to 1. The corresponding flags are available in POSEDGEIF and NEGEDGEIF. If none of the interrupts are enabled, the edge detection is disabled and POSEDGEIF and NEGEDGEIF hold their last value until cleared or set by software. Disabling the edge detection is only allowed on NEGEDGEIF. Both flags act as EM2 wakeup sources if the corresponding IEN is set high.

9.5.3.5 Clock Failure

In case the oscillator or crystal stops or does not output clock when expected, a failure interrupt can be raised. The failure occurs if fewer than 3 LFXO clock positive edges happen during one 1ms. The failure detection is enabled by setting FAILDETEN to 1 in LFXO_CTRL register. This bit acts as a clock requester. Once enabled, failure detection status can be checked by reading FAILIF in LFXO_IF register. If FAILIEN is set high, failure will generate both interrupt and EM2 wakeup. Failure detection is also implemented as EM4 wakeup source. To wakeup from EM4 on LFXO failure detection, set FAILDETEM4WUEN high in LFXO_CTRL.

9.5.3.6 Automatic Gain Control

AGC and HIGHAMPL in LFXO_CFG are settings applied to the LFXO oscillator. Both settings provide higher crystal oscillation amplitude. This will improve duty cycle in the output clock and give lower sensitivity to noise, but at the cost of higher current consumption. The AGC bit is used to enable the Automatic Gain Control module that adjusts the amplitude of the oscillations. It is enabled by default. When disabled, the LFXO will run at the start-up current and the crystal will oscillate rail-to-rail or limited by the start-up current. The HIGHAMPL bit will have no effect when AGC is disabled. When AGC is enabled setting the HIGHAMPL bit will give about 70% higher crystal oscillation amplitude.

9.5.3.7 Force Off

It is not allowed to write to LFXO_CFG unless LFXO is in FORCEOFF mode. If this guideline is violated, the write access is blocked and a bus fault is generated. Writing to CFG registers has no effect in DIGEXTCLK mode. Note: when putting the oscillators to FORCEOFF mode, wait for ENS status to go low for the oscillator to completely shut off. Once the oscillator is forced off, it is safe to write to the LFXO_CFG register.

9.5.3.8 Register Synchronization

While the CFG registers are static LFXO configuration, LFXO_CAL register has GAIN and CAPTUNE bit fields which can be written to while the oscillator is running. This is used to calibrate the LFXO clock. These registers are allowed to be written only if CALBSY in LFXO_SYNCBUSY register is low. If this guideline is violated, the write access is blocked and a bus fault is generated. CALBSY is guaranteed to be low in FORCEOFF mode. When exiting FORCEOFF mode, CALBSY will go high and stay high until the initial internal synchronization is done. CALBSY is also guaranteed to be low in DIGEXTCLK mode since writing to CAL register has no effect in DIGEXTCLK mode. CAPTUNE is allowed to be incremented or decremented by one LSB when not in FORCEOFF mode. Note that CAPTUNE tunes the internal capacitors connected to LFXTAL_I and LFXTAL_O pads (see Register map for more details). By programming GAIN bit field it is possible to optimize start-up time and power consumption for a given crystal. Internal capacitances are not provided on all chips (see the device data sheet for more details).

9.5.3.9 Register Lock

See the LFXO_LOCK register on how to lock certain registers. Registers LFXO_CTRL, LFXO_CFG, and LFXO_CAL are lockable. The LOCK bit in LFXO_STATUS register is available to check whether the registers are locked. If locked, all updates to these registers are blocked and bus faults are issued.

9.5.3.10 Reset Behavior

Upon reset, the LFXO is configured for the safe crystal start-up. The TIMEOUT is set to 32k cycles, The MODE is set to XTAL and the reset state is FORCEOFF. In order to minimize the start-up time and power consumption for a given crystal, it is possible to adjust the start-up gain in the oscillator by programming GAIN in LFXO_CAL. All controls are retained in EM4, except LFXO_IEN register which is reset after EM4 wakeup.

9.5.4 LFXO Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	LFXO_IPVERSION	R	LFXO IP version
0x004	LFXO_CTRL	RW	LFXO Control Register
0x008	LFXO_CFG	RW	LFXO Configuration Register
0x010	LFXO_STATUS	RH	LFXO Status Register
0x014	LFXO_CAL	RW LFSYNC	LFXO Calibration Register
0x018	LFXO_IF	RWH INTFLAG	Interrupt Flag Register
0x01C	LFXO_IEN	RW	Interrupt Enable Register
0x020	LFXO_SYNCBUSY	RH	LFXO Sync Busy Register
0x024	LFXO_LOCK	W	Configuration Lock Register
0x1000	LFXO_IPVERSION_SET	R	LFXO IP version
0x1004	LFXO_CTRL_SET	RW	LFXO Control Register
0x1008	LFXO_CFG_SET	RW	LFXO Configuration Register
0x1010	LFXO_STATUS_SET	RH	LFXO Status Register
0x1014	LFXO_CAL_SET	RW LFSYNC	LFXO Calibration Register
0x1018	LFXO_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x101C	LFXO_IEN_SET	RW	Interrupt Enable Register
0x1020	LFXO_SYNCBUSY_SET	RH	LFXO Sync Busy Register
0x1024	LFXO_LOCK_SET	W	Configuration Lock Register
0x2000	LFXO_IPVERSION_CLR	R	LFXO IP version
0x2004	LFXO_CTRL_CLR	RW	LFXO Control Register
0x2008	LFXO_CFG_CLR	RW	LFXO Configuration Register
0x2010	LFXO_STATUS_CLR	RH	LFXO Status Register
0x2014	LFXO_CAL_CLR	RW LFSYNC	LFXO Calibration Register
0x2018	LFXO_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x201C	LFXO_IEN_CLR	RW	Interrupt Enable Register
0x2020	LFXO_SYNCBUSY_CLR	RH	LFXO Sync Busy Register
0x2024	LFXO_LOCK_CLR	W	Configuration Lock Register
0x3000	LFXO_IPVERSION_TGL	R	LFXO IP version
0x3004	LFXO_CTRL_TGL	RW	LFXO Control Register
0x3008	LFXO_CFG_TGL	RW	LFXO Configuration Register
0x3010	LFXO_STATUS_TGL	RH	LFXO Status Register
0x3014	LFXO_CAL_TGL	RW LFSYNC	LFXO Calibration Register
0x3018	LFXO_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x301C	LFXO_IEN_TGL	RW	Interrupt Enable Register
0x3020	LFXO_SYNCBUSY_TGL	RH	LFXO Sync Busy Register

Offset	Name	Type	Description
0x3024	LFXO_LOCK_TGL	W	Configuration Lock Register

9.5.5 LFXO Register Description

9.5.5.1 LFXO_IPVERSION - LFXO IP version

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP Version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

9.5.5.2 LFXO_CTRL - LFXO Control Register

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0			0x1	0x0
Access																											RW	RW			RW	RW
Name																											FAILDETEM4WUEN	FAILDETEN			DISONDEMAND	FORCEEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	FAILDETEM4WUEN	0x0	RW	**LFXO Failure Detection EM4WU Enable**
	Set this bit to enable EM4 exit on the oscillator failure detection.			
4	FAILDETEN	0x0	RW	**LFXO Failure Detection Enable**
	Set this bit to enable the oscillator failure detection feature. Note that setting this bit will enable the oscillator core.			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	DISONDEMAND	0x1	RW	**LFXO Disable On-demand requests**
	Set this bit to disable On-demand requests.			
0	FORCEEN	0x0	RW	**LFXO Force Enable**
	Set this bit to enable the oscillator core. The oscillator core is enabled regardless of On-demand requests.			

9.5.5.3 LFXO_CFG - LFXO Configuration Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x7					0x0				0x0	0x1
Access																						RW					RW				RW	RW
Name																						TIMEOUT					MODE				HIGHAMPL	AGC

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	TIMEOUT	0x7	RW	**LFXO Start-up Delay**

Configures the start-up delay for LFXO.

Value	Mode	Description
0	CYCLES2	Timeout period of 2 cycles
1	CYCLES256	Timeout period of 256 cycles
2	CYCLES1K	Timeout period of 1024 cycles
3	CYCLES2K	Timeout period of 2048 cycles
4	CYCLES4K	Timeout period of 4096 cycles
5	CYCLES8K	Timeout period of 8192 cycles
6	CYCLES16K	Timeout period of 16384 cycles
7	CYCLES32K	Timeout period of 32768 cycles

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	MODE	0x0	RW	**LFXO Mode**

Selects the LFXO mode.

Value	Mode	Description
0	XTAL	A 32768Hz crystal should be connected to the LF crystal pads. Voltage must not exceed VDDIO.
1	BUFEXTCLK	An external sine source with minimum amplitude 100mv (zero-to-peak) and maximum amplitude 500mV (zero-to-peak) should be connected in series with LFXTAL_I pin. Minimum voltage should be larger than ground and maximum voltage smaller than VDDIO. The sine source does not need to be ac coupled externally as it is ac couples inside LFXO. LFXTAL_O is free to be used as a general purpose GPIO.
2	DIGEXTCLK	An external 32KHz CMOS clock should be provided on LFXTAL_I. LFXTAL_O is free to be used as a general purpose GPIO.

Bit	Name	Reset	Access	Description
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	HIGHAMPL	0x0	RW	**LFXO High Amplitude Enable**
	Set this bit to enable high XTAL oscillation amplitude.			
0	AGC	0x1	RW	**LFXO AGC Enable**
	Set this bit to enable automatic gain control which limits XTAL oscillation amplitude.			

9.5.5.4 LFXO_STATUS - LFXO Status Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0															0x0																0x0
Access	R															R															R
Name	LOCK															ENS																RDY

Bit	Name	Reset	Access	Description
31	LOCK	0x0	R	**LFXO Locked Status**
	If set, all LFXO lockable registers are locked.			

Value	Mode	Description
0	UNLOCKED	LFXO lockable registers are not locked
1	LOCKED	LFXO lockable registers are locked

30:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	ENS	0x0	R	**LFXO Enable Status**
	LFXO is enabled.			
15:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	RDY	0x0	R	**LFXO Ready Status**
	LFXO is enabled and start-up time has exceeded.			

9.5.5.5 LFXO_CAL - LFXO Calibration Register

Offset	Bit Position																															
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x2						0x0			
Access																							RW						RW			
Name																							GAIN						CAPTUNE			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	GAIN	0x2	RW	**LFXO Startup Gain** The optimal value depends on the chosen crystal.
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	CAPTUNE	0x0	RW	**Internal Capacitance Tuning** Program internal load capacitance connected between X_N pin and ground and X_P pin and ground. The bus affects tuning capacitances on both pins symmetrically. CAPTUNE value must not exceed 0x4F. When updating CAPTUNE, its value must only be incremented or decremented by 1 which provides a tuning step of 0.25pF. The maximum value is estimated to be 20pF. Please refer to the device Datasheet for more information.

9.5.5.6 LFXO_IF - Interrupt Flag Register

Offset	Bit Position																															
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0	0x0	0x0
Access																													RW	RW	RW	RW
Name																													FAIL	NEGEDGE	POSEDGE	RDY

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	FAIL	0x0	RW	**LFXO Failure Interrupt Flag**
	Set when LFXO failure is detected. Write 1 to clear the interrupt flag.			
2	NEGEDGE	0x0	RW	**Falling Edge Interrupt Flag**
	Triggers on every negative edge of the LFXO clock.			
1	POSEDGE	0x0	RW	**Rising Edge Interrupt Flag**
	Triggers on every positive edge of the LFXO clock.			
0	RDY	0x0	RW	**LFXO Ready Interrupt Flag**
	Set when LFXO is ready (start-up time exceeded). Write 1 to clear the interrupt flag.			

9.5.5.7 LFXO_IEN - Interrupt Enable Register

Offset	Bit Position																														
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0	0x0	0x0
Access																													RW	RW	RW	RW
Name																													FAIL	NEGEDGE	POSEDGE	RDY

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	FAIL	0x0	RW	**LFXO Failure Interrupt Enable**
	Write 1 to enable FAILIF			
2	NEGEDGE	0x0	RW	**Falling Edge Interrupt Enable**
	Write 1 to enable NEGEDGEIF.			
1	POSEDGE	0x0	RW	**Rising Edge Interrupt Enable**
	Write 1 to enable POSEDGEIF.			
0	RDY	0x0	RW	**LFXO Ready Interrupt Enable**
	Write 1 to enable RDYIF.			

9.5.5.8 LFXO_SYNCBUSY - LFXO Sync Busy Register

Offset	Bit Position																														
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																R
Name																																CAL

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CAL	0x0	R	**LFXO Synchronization status**
	This bit is set when there is an ongoing synchronization of CAL register bitfields. Do not write to CAL register while this bit is set.			

9.5.5.9 LFXO_LOCK - Configuration Lock Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x1A20								
Access																								W								
Name																								LOCKKEY								

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x1A20	W	**Lock Key** Write any other value than UNLOCK to lock CTRL, CFG and CAL registers. Write UNLOCK value to unlock the registers.

Value	Mode	Description
6688	UNLOCK	Unlock LFXO lockable registers

9.6 LFRCO - Low-Frequency RC Oscillator

9.6.1 Introduction

The LFRCO is an integrated low-frequency (32.768 kHz) RC oscillator. It can be used as a timing reference in EM0, EM1, EM2, and EM4. On certain part numbers a precision mode is available in EM0, EM1 and EM2. Precision mode enables hardware that periodically recalibrates the LFRCO against the HFXO crystal frequency when temperature changes to provide a fully interenal 32.768 kHz clock source with +/-500 ppm accuracy. Consult the device data sheet for details on which part numbers support precision mode.

9.6.2 Features

- 32.768 kHz oscillator
- High Accuracy
- Precision mode available in EM0, EM1, and EM2.
- Low-power non-precision operation in EM0, EM1, EM2, and EM4.
- On-demand
- Lockable registers

9.6.3 Functional Description

9.6.3.1 Start-up

The LFRCO has a fast start-up time. Please refer to the data sheet electrical specifications for the exact start-up time. When the oscillator has started up and is ready to use, the RDY status bit will go high and the RDY interrupt will be triggered. After startup, it may take two cycles for the clock to propagate through the CMU to the peripherals.

9.6.3.2 On-Demand Clocking

Hardware can request to enable the LFRCO by setting the LFRCO_STATUS.HWREQ bit field. The LFRCO can also optionally be configured via the LFRCO_STATUS.DISONDEMAND to shut down when no hardware request is present. This is known as on-demand clocking and allows the oscillator to be controlled without any software intervention.

9.6.3.3 Register Lock

The LFRCO configuration registers NOMCAL, NOMCALINV and CFG can be locked or unlocked by software using the LOCKKEY field in the LFRCO_LOCK register. By writing the UNLOCK value to LFRCO_LOCK_LOCKKEY, these registers will be unlocked and accessible to software. Any other value written to LFRCO_LOCK_LOCKKEY will lock the registers against write operations.

9.6.3.4 Precision Mode

Certain device families support a precision mode to bring the LFRCO accuracy to within +/-500 ppm, suitable for BLE sleep applications. Precision mode uses hardware to automatically re-calibrate the LFRCO against a crystal driven by the HFXO. Hardware detects temperature changes and initiates a re-calibration of the LFRCO as needed when operating in EM0, EM1, or EM2. If a re-calibration is necessary and the HFXO is not active, the precision mode hardware will automatilcally enable HFXO for a short time to perform the calibration. EM4 operation is not allowed while precision mode is enabled.

To enable precision mode software should set the LFRCO_CFG_HIGHPRECEN bit to 1 while the LFRCO is disabled (LFRCO_STATUS_ENS = 0). If this bit is written while the oscillator is enabled, a bus fault will be generated. In a typical application, software will only access HIGHPRECEN at startup before any peripherals are configured to request the LFRCO.

Disabling precision mode is the inverse - software should ensure that no peripherals are requesting the LFRCO, and then clear the LFRCO_CFG_HIGHPRECEN bit to 0.

9.6.3.4.1 Reference Frequency

Precision mode uses a reference clock from the HFXO as a calibration target. By default, a 38.4 MHz crystal clock source is assumed. 38.4 MHz is the only frequency supported by many of Silicon Labs' RF software stacks, and the default value should not normally be changed. Precision mode does support operation with different frequency crystals, however. Two registers (LFRCO_NOMCALCNT and LFRCO_NOMCALCNTINV) are used to specify the nominal relationship between the reference clock and the LFRCO frequency as shown in Figure 9.2 LFRCO_NOMCALCNT Calculation on page 214 and Figure 9.3 LFRCO_NOMCALCNTINV Calculation on page 214. NOMCALCNT and NOMCALCNTINV must be set up while the LFRCO is not enabled.

$$NOMCALCNT = (320 * f_{HFXO}) / 32768$$

Where f_{HFXO} is the reference crystal frequency in Hz

Figure 9.2. LFRCO_NOMCALCNT Calculation

$$NOMCALCNTINV = (1 / NOMCALCNT) * 2^{33}$$

Figure 9.3. LFRCO_NOMCALCNTINV Calculation

9.6.3.4.2 Temperature Check and Calibration Intervals

When starting up in precision mode, the LFRCO will calibrate itself against the HFXO. After startup, the die temperature is checked periodically and if necessary, the LFRCO will initiate a re-calibration. Periodic re-calibration is also performed if the temperature has not changed for an extended period of time.

If a very high die temperature gradient is expected (e.g. due to high-power RF transmission), software can reduce the temperature check interval temporarily by setting the LFRCO_CMD_REDUCETCINT bit to 1. The LFRCO will then use a shorter temperature check interval until the chip enters EM2. When EM2 is entered, the temperature-check interval will gradually be increased based on the measured temperature gradient until it is back to the normal temperature check interval.

9.6.3.5 Interrupts

The LFRCO implements several interrupt flags in the LFRCO_IF register to report status, error events, or for debugging. Each interrupt flag has an enable bit in LFRCO_IEN. Setting a bit in IEN to 1 enables the corresponding interrupt source to trigger an LFRCO interrupt.

RDYIF is triggered after start-up, when the LFRCO startup sequence is complete and the oscillator is ready to use.

POSEDGEIF and NEGEDGEIF are triggered by the rising and falling edge of LFRCO respectively. These flags will only get set if either of the interrupts are enabled (with POSEDGEIEN or NEGEDGEIEN). Note that enabling NEGEDGEIF or POSEDGEIF act as a clock requester for the LFRCO oscillator, and these two interrupt enables must be disabled in order to disable the LFRCO.

RDYIF, POSEDGEIF, and NEGEDGEIF are available only in EM0 and EM1.

Three of the interrupt sources are error flags used by precision mode to alert software that there is a problem with the oscillator and it may no longer be running with full precision. CALOORIF indicates that a calibration was performed, but the result was outside of the oscillator's adjustment range. TCOORIF indicates that a temperature measurement was performed, but the measurement was out of range. Finally, the SCHEDERRIF flag indicates that a temperature check could not be performed due to a prior error. These three error flags are available when operating in precision mode in EM0, EM1, or EM2.

The remaining flags, TEMPCHANGEIF, CALDONEIF and TCDONEIF are available when operating in precision mode in EM0 and EM1 only and are provided for debugging purposes. Refer to the LFRCO_IF register description for the specific conditions of these flags.

9.6.4 LFRCO Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	LFRCO_IPVERSION	R	IP version
0x004	LFRCO_CTRL	RW	Control Register
0x008	LFRCO_STATUS	RH	Status Register
0x014	LFRCO_IF	RWH INTFLAG	Interrupt Flag Register
0x018	LFRCO_IEN	RW	Interrupt Enable Register
0x020	LFRCO_LOCK	W	Configuration Lock Register
0x024	LFRCO_CFG	RW CONFIG	Configuration Register
0x02C	LFRCO_NOMCAL	RW CONFIG	Nominal Calibration Register
0x030	LFRCO_NOMCALINV	RW CONFIG	Nominal Calibration Inverted Register
0x034	LFRCO_CMD	W	Command Register
0x1000	LFRCO_IPVERSION_SET	R	IP version
0x1004	LFRCO_CTRL_SET	RW	Control Register
0x1008	LFRCO_STATUS_SET	RH	Status Register
0x1014	LFRCO_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1018	LFRCO_IEN_SET	RW	Interrupt Enable Register
0x1020	LFRCO_LOCK_SET	W	Configuration Lock Register
0x1024	LFRCO_CFG_SET	RW CONFIG	Configuration Register
0x102C	LFRCO_NOMCAL_SET	RW CONFIG	Nominal Calibration Register
0x1030	LFRCO_NOMCALINV_SET	RW CONFIG	Nominal Calibration Inverted Register
0x1034	LFRCO_CMD_SET	W	Command Register
0x2000	LFRCO_IPVERSION_CLR	R	IP version
0x2004	LFRCO_CTRL_CLR	RW	Control Register
0x2008	LFRCO_STATUS_CLR	RH	Status Register
0x2014	LFRCO_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2018	LFRCO_IEN_CLR	RW	Interrupt Enable Register
0x2020	LFRCO_LOCK_CLR	W	Configuration Lock Register
0x2024	LFRCO_CFG_CLR	RW CONFIG	Configuration Register
0x202C	LFRCO_NOMCAL_CLR	RW CONFIG	Nominal Calibration Register
0x2030	LFRCO_NOMCALINV_CLR	RW CONFIG	Nominal Calibration Inverted Register
0x2034	LFRCO_CMD_CLR	W	Command Register
0x3000	LFRCO_IPVERSION_TGL	R	IP version
0x3004	LFRCO_CTRL_TGL	RW	Control Register
0x3008	LFRCO_STATUS_TGL	RH	Status Register
0x3014	LFRCO_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3018	LFRCO_IEN_TGL	RW	Interrupt Enable Register

Offset	Name	Type	Description
0x3020	LFRCO_LOCK_TGL	W	Configuration Lock Register
0x3024	LFRCO_CFG_TGL	RW CONFIG	Configuration Register
0x302C	LFRCO_NOMCAL_TGL	RW CONFIG	Nominal Calibration Register
0x3030	LFRCO_NOMCALINV_TGL	RW CONFIG	Nominal Calibration Inverted Register
0x3034	LFRCO_CMD_TGL	W	Command Register

9.6.5 LFRCO Register Description

9.6.5.1 LFRCO_IPVERSION - IP version

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x1	R	**IP version ID**
	Reading this register returns the ip version number of LFRCO.			

9.6.5.2 LFRCO_CTRL - Control Register

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															DISONDEMAND	FORCEEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	DISONDEMAND	0x0	RW	**Disable On-Demand**
	Disable on demand functionality			
0	FORCEEN	0x0	RW	**Force Enable**
	Force the LFRCO core on			

9.6.5.3 LFRCO_STATUS - Status Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0															0x0																0x0
Access	R															R																R
Name	LOCK															ENS																RDY

Bit	Name	Reset	Access	Description
31	LOCK	0x0	R	**Lock Status**
	This bit is set when LFRCO is locked.			

Value	Mode	Description
0	UNLOCKED	Access to configuration registers not locked
1	LOCKED	Access to configuration registers locked

30:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	ENS	0x0	R	**Enabled Status**
	This bit is set when LFRCO is enabling the analog core.			
15:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	RDY	0x0	R	**Ready Status**
	This bit is set when qualification is done and LFRCO is ready.			

9.6.5.4 LFRCO_IF - Interrupt Flag Register

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0x0	0x0	0x0						0x0	0x0	0x0						0x0	0x0	0x0
Access														RW	RW	RW						RW	RW	RW						RW	RW	RW
Name														CALOOR	TCOOR	SCHEDERR						TEMPCHANGE	CALDONE	TCDONE						NEGEDGE	POSEDGE	RDY

Bit	Name	Reset	Access	Description
31:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	CALOOR	0x0	RW	**Calibration Out Of Range Flag**
	Triggers if Calibration measure is out of range			
17	TCOOR	0x0	RW	**Temperature Check Out Of Range Flag**
	Triggers if Temperature Check measure is out of range			
16	SCHEDERR	0x0	RW	**Scheduling Error Flag**
	Triggers if a scheduled Temperature Check can not be handeled because prior Termperature Check or Calibration did not complete.			
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	TEMPCHANGE	0x0	RW	**Temperature Change Flag**
	Triggers when Temperature Check detects a change in temperature			
9	CALDONE	0x0	RW	**Calibration Done Flag**
	Triggers on completion of Calibration			
8	TCDONE	0x0	RW	**Temperature Check Done Flag**
	Triggers on completion of Temperature Check			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	NEGEDGE	0x0	RW	**Falling Edge Flag**
	Triggers on every negative edge of the LFRCO clock. IF will only be set when corresponding IEN is set.			
1	POSEDGE	0x0	RW	**Rising Edge Flag**
	Triggers on every positive edge of the LFRCO clock. IF will only be set when corresponding IEN is set.			
0	RDY	0x0	RW	**Ready Flag**
	Triggers when the oscillator becomes ready			

9.6.5.5 LFRCO_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0x0	0x0	0x0						0x0	0x0	0x0						0x0	0x0	0x0
Access														RW	RW	RW						RW	RW	RW						RW	RW	RW
Name														CALOOR	TCOOR	SCHEDERR						TEMPCHANGE	CALDONE	TCDONE						NEGEDGE	POSEDGE	RDY

Bit	Name	Reset	Access	Description
31:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	CALOOR	0x0	RW	**Calibration Out Of Range Enable**
	Enables the Calibration Out Of Range interrupt			
17	TCOOR	0x0	RW	**Temperature Check Out Of Range Enable**
	Enables the Temperature Check Out Of Range interrupt			
16	SCHEDERR	0x0	RW	**Scheduling Error Enable**
	Enables the Scheduling Error interrupt			
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	TEMPCHANGE	0x0	RW	**Temperature Change Enable**
	Enables the Temperature Change interrupt			
9	CALDONE	0x0	RW	**Calibration Done Enable**
	Enables the Calibration Done interrupt			
8	TCDONE	0x0	RW	**Temperature Check Done Enable**
	Enables the Temperature Check Done interrupt			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	NEGEDGE	0x0	RW	**Falling Edge Enable**
	Enables the negedge interrupt and will cause the oscillator to run			
1	POSEDGE	0x0	RW	**Rising Edge Enable**
	Enables the posedge interrupt and will cause the oscillator to run			
0	RDY	0x0	RW	**Ready Enable**
	Enables the ready interrupt			

9.6.5.6 LFRCO_LOCK - Configuration Lock Register

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									LOCKKEY							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x0	W	**Lock Key**

Writing the lock key will unlock the LFRCO configuration registers. Writing any other value will lock them.

Value	Mode	Description
0	LOCK	Lock Configuration Registers
3987	UNLOCK	Unlock Configuration Registers

9.6.5.7 LFRCO_CFG - Configuration Register

Offset	Bit Position																															
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																HIGHPRECEN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	HIGHPRECEN	0x0	RW	**High Precision Enable**

LFRCO operates in High Precision Mode when this bit is set. HIGHPRECEN should not be written while LFRCO is enabled.

9.6.5.8 LFRCO_NOMCAL - Nominal Calibration Register

Offset	Bit Position																															
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x5B8D8										
Access																						RW										
Name																						NOMCALCNT										

Bit	Name	Reset	Access	Description
31:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20:0	NOMCALCNT	0x5B8D8	RW	**Nominal Calibration Count**
	Expected Calibration count value. Should be set based on HFXO frequency; NOMCALCNT = 2 * f(HFXO) / (32.768 kHz / 160). Default value corresponds to 38.4 MHz HFXO frequency. NOMCALCNT should not be written while LFRCO is enabled.			

9.6.5.9 LFRCO_NOMCALINV - Nominal Calibration Inverted Register

Offset	Bit Position																															
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x597A								
Access																								RW								
Name																								NOMCALCNTINV								

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16:0	NOMCALCNTINV	0x597A	RW	**Nominal Calibration Count Inverted**
	This register should always be set to the inverse of NOMCALCNT value. Due to format of this register, integer value should be NOMCALCNTINV = (1 / NOMCALCNT) * (2**33). NOMCALCNTINV should not be written while LFRCO is enabled.			

9.6.5.10 LFRCO_CMD - Command Register

Offset	Bit Position																															
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																W
Name																																REDUCETCINT

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	REDUCETCINT	0x0	W	**Reduce Temperature Check Interval** Setting this register field will temporarily lower the Temperature Check interval.

9.7 FSRCO - Fast Start RCO

9.7.1 Introduction

This is an RC oscillator which can start and stop very fast. It is a fixed frequency oscillator, with no frequency configurability and as such any user of this clock can rely on it being a specific frequency independent of the system state. This is the first oscillator used during power up and hence it minimizes dependency to other blocks.

9.7.2 Features

- 20 MHz nominal frequency
- Low energy consumption

9.7.3 Functional Description

There are no programmable registers in this module. Software can choose to use this as system clock in the CMU block. the only way to enable or disable the FSRCO is by requesting it as a clock source in the CMU clock select registers.

9.7.4 FSRCO Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	FSRCO_IPVERSION	R	IP Version
0x1000	FSRCO_IPVERSION_SET	R	IP Version
0x2000	FSRCO_IPVERSION_CLR	R	IP Version
0x3000	FSRCO_IPVERSION_TGL	R	IP Version

9.7.5 FSRCO Register Description

9.7.5.1 FSRCO_IPVERSION - IP Version

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP Version**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

9.8 ULFRCO - Ultra Low Frequency RC Oscillator

9.8.1 Introduction

The ULFRCO is a ultra low power 1 kHz oscillator which is always on in all energy modes except EM4. The ULFRCO is available to many low-frequency peripherals as a lower power alternative to one of the 32 kHz oscillators. This oscillator is also used for internal bias and housekeeping tasks.

9.8.2 Features

- 1 kHz nominal frequency
- Low energy consumption

9.8.3 Functional Description

There are no user programmable registers in this module. The oscillator will stop on EM4 entry and restart automatically on EM4 exit.

10. SMU - Security Management Unit

Quick Facts

What?

The Security Management Unit (SMU) provides configuration and status reporting for ARM TrustZone on the EFR32xG22.

Why?

Enables a robust solution at the system level.

How?

Hardware context switching and enhanced security provided by ARM TrustZone. Extension of the ARM MPU to control peripheral access.

10.1 Introduction

The Security Management Unit is used to configure and extend TrustZone bus level security provided by the Cortex-M33. In addition it increases the effective MPU regions by providing MPU control over peripheral access.

10.2 Features

- Per peripheral privileged and secure attributes
- Per master privileged and secure attributes
- Separate interrupt flags for privileged, secure, or instruction access exceptions.
- Separate interrupt flag for secure master access exceptions
- Secure and Privileged exception IRQs
- Configurable secure, non-secure, and non-secure-callable memory regions.

10.3 Functional Description

10.3.1 Bus Level Security

Bus level security is the ability to control the flow of information on the device. The components of bus level security are the Cortex-M33, the Bus Master Protect Unit (BMPU), and the Peripheral Protection Unit (PPU) as highlighted in Figure 10.1 Bus Level Security Implementation on page 227. The SMU controls and configures all the components used in bus level security.

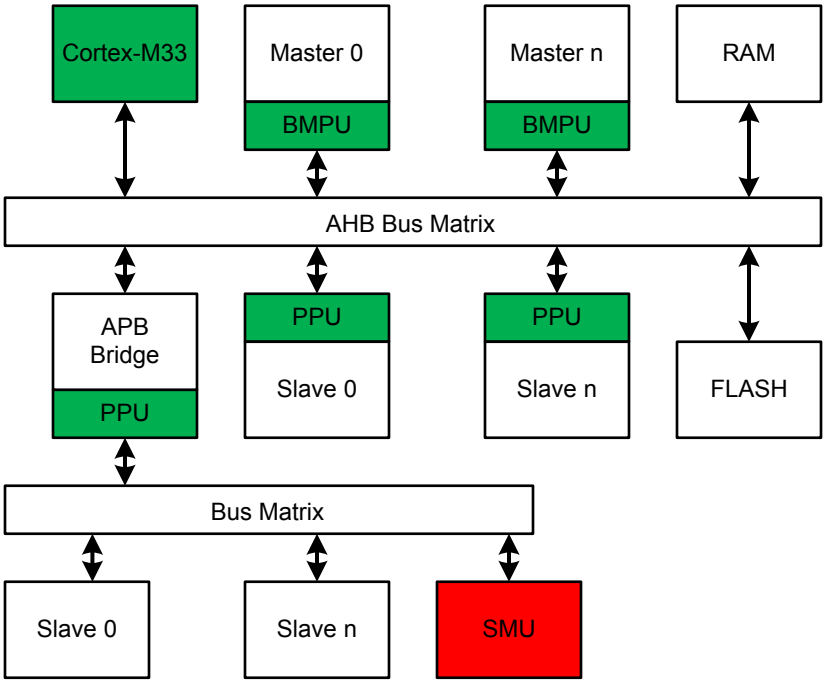


Figure 10.1. Bus Level Security Implementation

The BMPU is responsible for preventing masters (CPU, DMA, Etc..) from accessing secure addresses without authorization. For example, if a DMA configured as non-secure tries to access memory that is marked secure the BMPU will prevent access and set the corresponding interrupt flag. The BMPU prevents access of secure addresses by non-secure masters. The Cortex-M33 has BMPU functionality built into the TrustZone implementation.

The PPU is primarily responsible for blocking access to privileged peripherals from unprivileged masters. In addition, it also ensures that secure and non-secure peripherals are only accessible at the appropriate secure or non-secure addresses as described in 10.3.6 Configuring Peripherals.

Since FLASH and RAM have no PPU, bus masters of any privilege state may access those resources. The Cortex-M33 has an MPU which prevents execution of privileged memory when the CPU is in an unprivileged state. For more information on the MPU refer to the ARM Cortex-M33 documentation.

10.3.2 Privileged Access Control

The Cortex-M33 and all other masters can be in either the privileged or unprivileged state. All bus access to peripherals are tested for privilege level by the PPU and resolved as shown in Table 10.1 Privileged Access Table on page 228.

If an exception is detected on a write, the write will be ignored and the appropriate interrupt flag set. If an exception is detected on a read 0x0 will be returned and the appropriate interrupt flag set.

Table 10.1. Privileged Access Table

Master Attribute	Peripheral Attribute	Result
privileged	privileged	Success
privileged	unprivileged	Success
unprivileged	privileged	Exception
unprivileged	unprivileged	Success

10.3.3 Secure Access Control

The Cortex-M33 and all other masters can be in either the secure or non-secure state. All bus accesses are tested for security status by the BMPUs and PPUs and resolve as shown in Table 10.2 Secure Access Table on page 228 Secure access is computed using the secure attribute of the master and the address region being accessed. If a peripheral is being accessed, the secure attribute of the peripheral is also used. For more information on the relationship between the address regions and peripheral security attributes please see 10.3.6 Configuring Peripherals

If an exception is detected on a write the write will be ignored and the appropriate interrupt flag set. If an exception is detected on a read 0x0 will be returned and the appropriate interrupt flag set.

Table 10.2. Secure Access Table

Master Attribute	Address Attribute	Peripheral Attribute	Result
secure	secure	N/A	Success
secure	secure	secure	Success
secure	secure	non-secure	Exception
secure	non-secure	N/A	Exception
secure	non-secure	secure	Exception
secure	non-secure	non-secure	Success
non-secure	secure	N/A	Exception
non-secure	secure	secure	Exception
non-secure	secure	non-secure	Exception
non-secure	non-secure	N/A	Success
non-secure	non-secure	secure	Exception
non-secure	non-secure	non-secure	Success

10.3.4 ARM TrustZone

ARM TrustZone is used to control what addresses are accessible by the CPU at any given time. There are two security states: secure and non-secure. In addition the MPU provides two privilege levels: privileged and unprivileged. This results in 4 possible states: secure-privileged, non-secure-privileged, secure-unprivileged and non-secure-unprivileged.

Non-secure code may not directly call secure code. To call secure code, non-secure code must first call a shim located in specially marked non-secure-callable memory. Unprivileged code may invoke privileged code and change the processor state to privileged by either issuing an SVC instruction or taking an interrupt. The processor is returned to unprivileged state when software manually reconfigures the security state or exits an interrupt.

For more information on secure/non-secure and privileged/unprivileged state transitions see the ARM Cortex-M33 documentation.

There are two primary use cases for TrustZone and the MPU. The first is simply partitioning a monolithic application in to the 4 states to protect some pieces of the system from bugs or attacks on others. The second is to use a RTOS to isolate several tasks from each other. In this case the RTOS itself normally consumes the privileged states with all other code running in the unprivileged states. Whenever a task switch occurs the RTOS can reconfigure the device so the new task has access to only the components it requires, protecting other tasks from interference.

In both use cases the TrustZone and MPU feature of the Cortex-M33 both secures and accelerates mode transitions while the SMU provides the ability to configure the security and privilege attributes of peripherals and memory.

The core is in secure-privileged state after a reset.

10.3.5 Configuring Masters

The SMU provides the ability to configure the current secure and privileged attribute of all bus masters except for the CPU which is controlled as described in 10.3.4 ARM TrustZone.

To configure the privileged attribute of a master set the appropriate bit in SMU_BMPUPATDn. To configure the secure attribute of a master set the appropriate bit in SMU_BMPUPSATDn.

10.3.6 Configuring Peripherals

The SMU provides the ability to configure the current secure and privileged state of all peripherals. To configure the privileged attribute of a peripheral set the appropriate bit in SMU_PPUPATDn.

Each peripheral is accessible at one of two addresses: A secure address and an non-secure address. Which address is valid depends on the security attribute of the peripheral configured in the SMU. When configured as secure a peripheral may only be accessed at its secure address and when configured as non-secure the peripheral may only be accessed at its non-secure address. This forces code to be aware of the security attribute of the peripheral being accessed, preventing secure code from accessing a non-secure peripheral unintentionally.

The device memory map contains 4 regions of fixed length and fixed security attribute to facilitate the secure access of peripherals and RF peripherals. There is one secure (0x40000000) and one non-secure (0x50000000) region for peripherals and one secure (0xA0000000) and non-secure (0xB0000000) region for the radio subsystem. While each peripheral can be configured independently the radio subsystem is configured as a unit.

To configure the security attribute of a peripheral set the appropriate bit in SMU_PPUSATDn.

10.3.7 Configuring Memory

The SMU provides the ability to configure the security attribute of memory. There are 13 configurable regions in total. There are three regions in FLASH (0 - 2) and three in RAM (4-6) which have pre-determined secure attributes and user selectable sizes. Regions 3 and 11 cover the flash info page and ARM EPPB space respectively and have a fixed size. These regions can be configured as secure or non-secure by setting ESAUR3NS in SMU_ESAURTYPES0 and ESAUR11NS in SMU_ESAURTYPES1 respectively.

The size of the FLASH and RAM regions are controlled by the SMU_ESAURMBRxy registers as shown in Table 10.3 Memory Configuration Regions on page 230. Region sizes are adjusted in 4 kB increments with the lower 12 bits of SMU_ESAURMBRxy ignored. The non-secure-callable regions may be set to size 0 but secure and non-secure regions must be at least 4 kB.

Table 10.3. Memory Configuration Regions

Region	Memory	Attributes	Start	End
0	FLASH	secure	0x00000000	SMU_ESAURMBR01
1	FLASH	non-secure-callable	SMU_ESAURMBR01	SMU_ESAURMBR12
2	FLASH	non-secure	SMU_ESAURMBR12	0x0FE00000
3	FLASH (info page)	secure or non-secure	0x0FE00000	0x0FFFFFFF
4	RAM	secure	0x20000000	SMU_ESAURMBR45
5	RAM	non-secure-callable	SMU_ESAURMBR45	SMU_ESAURMBR56
6	RAM	non-secure	SMU_ESAURMBR56	0x2FFFFFFF
7	Peripherals	secure	0x40000000	0x4FFFFFFF
8	Peripherals	non-secure	0x50000000	0x5FFFFFFF
9	SEQRAM/ FRCRAM	secure	0xA0000000	0xAFFFFFFF
10	SEQRAM/ FRCRAM	non-secure	0xB0000000	0xBFFFFFFF
11	EPPB	secure or non-secure	0xE0044000	0xE00FDFFF
12	Cortex-M33 Processor ROM table	exempt	0xE00FE000	0xE00FEFFF

10.3.8 Cortex-M33 Integration

In addition to the SMU based access controls the Cortex-M33 has additional security features for controlling both secure and privileged access.

The Security Attribution Unit (SAU) provides that ability to setup secure memory regions in addition to those configured by the SMU. To disable the SAU and rely entirely on the SMU for security management clear ENABLE and set ALLNS in the SAU CTRL register. To enable a combination of SMU and SAU control set ENABLE in the SAU CTRL register. If both ENABLE and ALLNS are cleared all Cortex-M33 will treat all transactions as secure.

When both SAU and SMU are in use, a memory address is considered secure if either the SAU or SMU have it configured as secure. When enabled the SAU applies ONLY to access by the Cortex-M33 and does not effect any other masters. For more information on the SAU refer to ARM documentation.

Note: It is highly recommended that systems avoid using the SAU unless necessary. Since the SAU does not affect any masters outside the Cortex-M33, extreme care must be taken to ensure the SAU regions can not be trivially by bypassed through use of another master such as the DMA.

In addition to the Cortex-M33 MPU provides the ability to control which regions of FLASH and RAM are marked as privileged and prevent execution of privileged code by a CPU in unprivileged state. For more information on the configuration and use of the MPU refer to ARM documentation.

10.3.9 Exception Handling

When a BMPU detects a non-secure master attempting to access a secure address, the BMPUSECIF in SMU_IF is set and the ID of the Master block is written to SMU_BMPUFS. If BMPUSECIEN is set and the SMU's Secure IRQ enabled, the CPU will be interrupted.

When a PPU detects an access to a secure peripheral at its non-secure address or an access to a non-secure peripheral at its secure address, PPUSECIF in SMU_IF is set and the ID of the peripheral being accessed is written to SMU_PPUFS. If PPUSECIEN is set and the SMU's Secure IRQ enabled, the CPU will be interrupted.

If a PPU detects an attempt to fetch an instruction from a peripheral, PPUINSTIF in SMU_IF will be set and the ID of the peripheral being accessed is written to SMU_PPUFS. If PPUINSTIEN is set and the SMU's Privileged IRQ enabled, the CPU will be interrupted.

If a PPU detects an attempt to access a privileged peripheral by an unprivileged master, PPUPRIVIF in SMU_IF will be set and the ID of the peripheral being accessed is written to SMU_PPUFS. If PPUPRIVIEN is set and the SMU's Privileged IRQ enabled, the CPU will be interrupted.

When any IRQ is trigged the Cortex-M33 is automatically placed in the privileged state. The security state is determined by configuration inside the Cortex-M33. Refer to ARM's documentation for more details.

If the SMU is configured in an inconsistent way, the SMUPRGERR flag in SMU_STATUS will be set. One example of an invalid configuration is setting SMU_ESAURMBR01 to a value larger than SMU_ESAURMBR23. SMUPRGERR should be checked after the SMU is configured.

10.3.10 SMU Lock

The SMU registers can be locked to prevent unintended modifications. SMULOCK in SMU_STATUS indicates if the SMU is currently locked. To unlock the SMU write 0xACCE55 to the SMU_LOCK register. To lock write any other value to SMU_LOCK.

In addition to locking the SMU registers the SMU can prevent access to the Cortex-M33 ASU, MPU, SMPU, VTOR and VTAIRCR registers. To lock access to one or more of these blocks set the corresponding bit in SMU_M33CTRL.

10.4 SMU Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	SMU_IPVERSION	R	IP Version
0x004	SMU_STATUS	RH	Status
0x008	SMU_LOCK	W	Lock
0x00C	SMU_IF	RWH INTFLAG	Interrupt Flag
0x010	SMU_IEN	RW	Interrupt Enable
0x020	SMU_M33CTRL	RW	M33 Control
0x040	SMU_PPUPATD0	RW	PPU PATD Register 0
0x044	SMU_PPUPATD1	RW	PPU PATD Register 1
0x060	SMU_PPUSATD0	RW	PPU SATD Register 0
0x064	SMU_PPUSATD1	RW	PPU SATD Register 1
0x140	SMU_PPUFS	RH	PPU Fault Status
0x150	SMU_BMPUPATD0	RW	BMPU PATD Register 0
0x170	SMU_BMPUSATD0	RW	BMPU SATD Register 0
0x250	SMU_BMPUFS	RH	BMPU Fault Status
0x254	SMU_BMPUFSADDR	RH	BMPU Fault Status Address
0x260	SMU_ESAURTYPES0	RW	ESAU Region Types Register 0
0x264	SMU_ESAURTYPES1	RW	ESAU Region Types Register 1
0x270	SMU_ESAUMRB01	RW	ESAU Movable Region Boundary 0-1
0x274	SMU_ESAUMRB12	RW	ESAU Movable Region Boundary 1-2
0x280	SMU_ESAUMRB45	RW	ESAU Movable Region Boundary 4-5
0x284	SMU_ESAUMRB56	RW	ESAU Movable Region Boundary 5-6
0x1000	SMU_IPVERSION_SET	R	IP Version
0x1004	SMU_STATUS_SET	RH	Status
0x1008	SMU_LOCK_SET	W	Lock
0x100C	SMU_IF_SET	RWH INTFLAG	Interrupt Flag
0x1010	SMU_IEN_SET	RW	Interrupt Enable
0x1020	SMU_M33CTRL_SET	RW	M33 Control
0x1040	SMU_PPUPATD0_SET	RW	PPU PATD Register 0
0x1044	SMU_PPUPATD1_SET	RW	PPU PATD Register 1
0x1060	SMU_PPUSATD0_SET	RW	PPU SATD Register 0
0x1064	SMU_PPUSATD1_SET	RW	PPU SATD Register 1
0x1140	SMU_PPUFS_SET	RH	PPU Fault Status
0x1150	SMU_BMPUPATD0_SET	RW	BMPU PATD Register 0
0x1170	SMU_BMPUSATD0_SET	RW	BMPU SATD Register 0
0x1250	SMU_BMPUFS_SET	RH	BMPU Fault Status

Offset	Name	Type	Description
0x1254	SMU_BMPUFSADDR_SET	RH	BMPU Fault Status Address
0x1260	SMU_ESAURTYPES0_SET	RW	ESAU Region Types Register 0
0x1264	SMU_ESAURTYPES1_SET	RW	ESAU Region Types Register 1
0x1270	SMU_ESAUMRB01_SET	RW	ESAU Movable Region Boundary 0-1
0x1274	SMU_ESAUMRB12_SET	RW	ESAU Movable Region Boundary 1-2
0x1280	SMU_ESAUMRB45_SET	RW	ESAU Movable Region Boundary 4-5
0x1284	SMU_ESAUMRB56_SET	RW	ESAU Movable Region Boundary 5-6
0x2000	SMU_IPVERSION_CLR	R	IP Version
0x2004	SMU_STATUS_CLR	RH	Status
0x2008	SMU_LOCK_CLR	W	Lock
0x200C	SMU_IF_CLR	RWH INTFLAG	Interrupt Flag
0x2010	SMU_IEN_CLR	RW	Interrupt Enable
0x2020	SMU_M33CTRL_CLR	RW	M33 Control
0x2040	SMU_PPUPATD0_CLR	RW	PPU PATD Register 0
0x2044	SMU_PPUPATD1_CLR	RW	PPU PATD Register 1
0x2060	SMU_PPUSATD0_CLR	RW	PPU SATD Register 0
0x2064	SMU_PPUSATD1_CLR	RW	PPU SATD Register 1
0x2140	SMU_PPUFS_CLR	RH	PPU Fault Status
0x2150	SMU_BMPUPATD0_CLR	RW	BMPU PATD Register 0
0x2170	SMU_BMPUSATD0_CLR	RW	BMPU SATD Register 0
0x2250	SMU_BMPUFS_CLR	RH	BMPU Fault Status
0x2254	SMU_BMPUFSADDR_CLR	RH	BMPU Fault Status Address
0x2260	SMU_ESAURTYPES0_CLR	RW	ESAU Region Types Register 0
0x2264	SMU_ESAURTYPES1_CLR	RW	ESAU Region Types Register 1
0x2270	SMU_ESAUMRB01_CLR	RW	ESAU Movable Region Boundary 0-1
0x2274	SMU_ESAUMRB12_CLR	RW	ESAU Movable Region Boundary 1-2
0x2280	SMU_ESAUMRB45_CLR	RW	ESAU Movable Region Boundary 4-5
0x2284	SMU_ESAUMRB56_CLR	RW	ESAU Movable Region Boundary 5-6
0x3000	SMU_IPVERSION_TGL	R	IP Version
0x3004	SMU_STATUS_TGL	RH	Status
0x3008	SMU_LOCK_TGL	W	Lock
0x300C	SMU_IF_TGL	RWH INTFLAG	Interrupt Flag
0x3010	SMU_IEN_TGL	RW	Interrupt Enable
0x3020	SMU_M33CTRL_TGL	RW	M33 Control
0x3040	SMU_PPUPATD0_TGL	RW	PPU PATD Register 0
0x3044	SMU_PPUPATD1_TGL	RW	PPU PATD Register 1
0x3060	SMU_PPUSATD0_TGL	RW	PPU SATD Register 0

Offset	Name	Type	Description
0x3064	SMU_PPUSATD1_TGL	RW	PPU SATD Register 1
0x3140	SMU_PPUFS_TGL	RH	PPU Fault Status
0x3150	SMU_BMPUPATD0_TGL	RW	BMPU PATD Register 0
0x3170	SMU_BMPUSATD0_TGL	RW	BMPU SATD Register 0
0x3250	SMU_BMPUFS_TGL	RH	BMPU Fault Status
0x3254	SMU_BMPUFSADDR_TGL	RH	BMPU Fault Status Address
0x3260	SMU_ESAURTYPES0_TGL	RW	ESAU Region Types Register 0
0x3264	SMU_ESAURTYPES1_TGL	RW	ESAU Region Types Register 1
0x3270	SMU_ESAUMRB01_TGL	RW	ESAU Movable Region Boundary 0-1
0x3274	SMU_ESAUMRB12_TGL	RW	ESAU Movable Region Boundary 1-2
0x3280	SMU_ESAUMRB45_TGL	RW	ESAU Movable Region Boundary 4-5
0x3284	SMU_ESAUMRB56_TGL	RW	ESAU Movable Region Boundary 5-6

10.5 SMU Register Description

10.5.1 SMU_IPVERSION - IP Version

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x1	R	**IP Version**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

10.5.2 SMU_STATUS - Status

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															R	R
Name																															SMUPRGERR	SMULOCK

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	SMUPRGERR	0x0	R	**SMU Programming Error**
	Indicates if SMU Registers were programmed incorrectly.			
0	SMULOCK	0x0	R	**SMU Lock**
	Indicates if SMU Registers are locked.			

Value	Mode	Description
0	UNLOCKED	SMULOCK is Unlocked
1	LOCKED	SMULOCK is Locked

10.5.3 SMU_LOCK - Lock

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0																							
Access									W																							
Name									SMULOCKKEY																							

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:0	SMULOCKKEY	0x0	W	**SMU Lock/Key**
	Write anything but UNLOCK to lock registers.			

Value	Mode	Description
11325013	UNLOCK	Unlocks Registers

10.5.4 SMU_IF - Interrupt Flag

Offset	Bit Position																														
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0	0x0														0x0		0x0
Access															RW	RW														RW		RW
Name															BMPUSEC	PPUSEC														PPUINST		PPUPRIV

Bit	Name	Reset	Access	Description
31:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	BMPUSEC	0x0	RW	**BMPU Security Interrupt Flag**
	Triggered when a security fault occurs in the Bus Master Protection Unit.			
16	PPUSEC	0x0	RW	**PPU Security Interrupt Flag**
	Triggered when a security fault occurs in the Peripheral Protection Unit.			
15:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	PPUINST	0x0	RW	**PPU Instruction Interrupt Flag**
	Triggered when a instruction fault occurs in the Peripheral Protection Unit.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	PPUPRIV	0x0	RW	**PPU Privilege Interrupt Flag**
	Triggered when a privilege fault occurs in the Peripheral Protection Unit.			

10.5.5 SMU_IEN - Interrupt Enable

Offset	Bit Position																														
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0	0x0														0x0		0x0
Access															RW	RW														RW		RW
Name															BMPUSEC	PPUSEC														PPUINST		PPUPRIV

Bit	Name	Reset	Access	Description
31:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	BMPUSEC	0x0	RW	**BMPU Security Interrupt Enable**
	Set to enable the BMPUSECIF Interrupt			
16	PPUSEC	0x0	RW	**PPU Security Interrupt Enable**
	Set to enable the PPUSECIF Interrupt.			
15:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	PPUINST	0x0	RW	**PPU Instruction Interrupt Enable**
	Set to enable the PPUINSTIF Interrupt.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	PPUPRIV	0x0	RW	**PPU Privilege Interrupt Enable**
	Set to enable the PPUPRIVIF Interrupt.			

10.5.6 SMU_M33CTRL - M33 Control

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												RW	RW	RW	RW	RW
Name																												LOCKSAU	LOCKNSMPU	LOCKSMPU	LOCKNSVTOR	LOCKSVTAIRCR

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	LOCKSAU	0x0	RW	**LOCKSAU control of M33 CPU**
	Set to 1 lock security attribution unit.			
3	LOCKNSMPU	0x0	RW	**LOCKNSMPU control of M33 CPU**
	Set to 1 lock non-secure MPU configuration.			
2	LOCKSMPU	0x0	RW	**LOCKSMPU control of M33 CPU**
	Set to 1 lock secure MPU configuration.			
1	LOCKNSVTOR	0x0	RW	**LOCKNSVTOR control of M33 CPU**
	Set to 1 lock non-secure VTOR.			
0	LOCKSVTAIRCR	0x0	RW	**LOCKSVTAIRCR control of M33 CPU**
	Set to 1 lock secure VTAIRCR.			

10.5.7 SMU_PPUPATD0 - PPU PATD Register 0

Offset	Bit Position																															
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	
Access	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
Name	DCI	GPCRC	IFADCDEBUG	BURAM	SYSCFG	SYSCFGCFGNS	CHIPTESTCTRL	I2C1	BURTC	USART1	USART0	TIMER4	TIMER3	TIMER2	TIMER1	TIMER0	LDMAXBAR	LDMA	GPIO	PRS	ICACHE0	MSC	ULFRCO	LFRCO	LFXO	DPLL0	FSRCO	HFRCO0	HFXO0	CMU	EMU	

Bit	Name	Reset	Access	Description
31	DCI	0x1	RW	**DCI Privileged Access**
	DCI Privileged Access			
30	GPCRC	0x1	RW	**GPCRC Privileged Access**
	GPCRC Privileged Access			
29	IFADCDEBUG	0x1	RW	**IFADCDEBUG Privileged Access**
	IFADCDEBUG Privileged Access			
28	BURAM	0x1	RW	**BURAM Privileged Access**
	BURAM Privileged Access			
27	SYSCFG	0x1	RW	**SYSCFG Privileged Access**
	SYSCFG Privileged Access			
26	SYSCFGCFGNS	0x1	RW	**SYSCFGCFGNS Privileged Access**
	SYSCFGCFGNS Privileged Access			
25	CHIPTESTCTRL	0x1	RW	**CHIPTESTCTRL Privileged Access**
	CHIPTESTCTRL Privileged Access			
24	I2C1	0x1	RW	**I2C1 Privileged Access**
	I2C1 Privileged Access			
23	BURTC	0x1	RW	**BURTC Privileged Access**
	BURTC Privileged Access			
22	USART1	0x1	RW	**USART1 Privileged Access**
	USART1 Privileged Access			
21	USART0	0x1	RW	**USART0 Privileged Access**
	USART0 Privileged Access			
20	TIMER4	0x1	RW	**TIMER4 Privileged Access**
	TIMER4 Privileged Access			
19	TIMER3	0x1	RW	**TIMER3 Privileged Access**
	TIMER3 Privileged Access			
18	TIMER2	0x1	RW	**TIMER2 Privileged Access**

Bit	Name	Reset	Access	Description
	TIMER2 Privileged Access			
17	TIMER1	0x1	RW	**TIMER1 Privileged Access**
	TIMER1 Privileged Access			
16	TIMER0	0x1	RW	**TIMER0 Privileged Access**
	TIMER0 Privileged Access			
15	LDMAXBAR	0x1	RW	**LDMAXBAR Privileged Access**
	LDMAXBAR Privileged Access			
14	LDMA	0x1	RW	**LDMA Privileged Access**
	LDMA Privileged Access			
13	GPIO	0x1	RW	**GPIO Privileged Access**
	GPIO Privileged Access			
12	PRS	0x1	RW	**PRS Privileged Access**
	PRS0 Privileged Access			
11	ICACHE0	0x1	RW	**ICACHE0 Privileged Access**
	ICACHE0 Privileged Access			
10	MSC	0x1	RW	**MSC Privileged Access**
	IMEM Privileged Access			
9	ULFRCO	0x1	RW	**ULFRCO Privileged Access**
	ULFRCO Privileged Access			
8	LFRCO	0x1	RW	**LFRCO Privileged Access**
	LFRCO Privileged Access			
7	LFXO	0x1	RW	**LFXO Privileged Access**
	LFXO Privileged Access			
6	DPLL0	0x1	RW	**DPLL0 Privileged Access**
	DPLL0 Privileged Access			
5	FSRCO	0x1	RW	**FSRCO Privileged Access**
	FSRCO Privileged Access			
4	HFRCO0	0x1	RW	**HFRCO0 Privileged Access**
	HFRCO0 Privileged Access			
3	HFXO0	0x1	RW	**HFXO0 Privileged Access**
	SYXO0 Privileged Access			
2	CMU	0x1	RW	**CMU Privileged Access**
	CMU Privileged Access			
1	EMU	0x1	RW	**EMU Privileged Access**
	EMU Privileged Access			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.8 SMU_PPUPATD1 - PPU PATD Register 1

Offset	Bit Position																													
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	
Access																	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
Name																	AHBRADIO	CRYPTOACC	EUART0	AMUXCP0	WDOG0	I2C0	IADC0	LETIMER0	RTCC	SMUCFGNS	SMU	RADIOAES	RFSENSE	PDM	DCDC	

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	AHBRADIO	0x1	RW	**AHBRADIO Privileged Access**
	AHBRADIO Privileged Access			
14	CRYPTOACC	0x1	RW	**CRYPTOACC Privileged Access**
	CRYPTOACC Privileged Access			
13	EUART0	0x1	RW	**EUART0 Privileged Access**
	EUART0 Privileged Access			
12	AMUXCP0	0x1	RW	**AMUXCP0 Privileged Access**
	AMUXCP0 Privileged Access			
11	WDOG0	0x1	RW	**WDOG0 Privileged Access**
	WDOG0 Privileged Access			
10	I2C0	0x1	RW	**I2C0 Privileged Access**
	I2C0 Privileged Access			
9	IADC0	0x1	RW	**IADC0 Privileged Access**
	IADC0 Privileged Access			
8	LETIMER0	0x1	RW	**LETIMER0 Privileged Access**
	LETIMER0 Privileged Access			
7	RTCC	0x1	RW	**RTCC Privileged Access**
	RTCC Privileged Access			
6	SMUCFGNS	0x1	RW	**SMUCFGNS Privileged Access**
	SMUCFGNS Privileged Access			
5	SMU	0x1	RW	**SMU Privileged Access**
	SMU Privileged Access			
4	RADIOAES	0x1	RW	**RADIOAES Privileged Access**
	RADIOAES Privileged Access			
3	RFSENSE	0x1	RW	**RFSENSE Privileged Access**
	RFSENSE Privileged Access			

Bit	Name	Reset	Access	Description
2	PDM PDM Privileged Access	0x1	RW	**PDM Privileged Access**
1	DCDC DCDC Privileged Access	0x1	RW	**DCDC Privileged Access**
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.9 SMU_PPUSATD0 - PPU SATD Register 0

Offset	Bit Position																															
0x060	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	
Access	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
Name	DCI	GPCRC	IFADCDEBUG	BURAM	SYSCFG	SYSCFGCFGNS	CHIPTESTCTRL	I2C1	BURTC	USART1	USART0	TIMER4	TIMER3	TIMER2	TIMER1	TIMER0	LDMAXBAR	LDMA	GPIO	PRS	ICACHE0	MSC	ULFRCO	LFRCO	LFXO	DPLL0	FSRCO	HFRCO0	HFXO0	CMU	EMU	

Bit	Name	Reset	Access	Description
31	DCI	0x1	RW	**DCI Secure Access**
	DCI Secure Access			
30	GPCRC	0x1	RW	**GPCRC Secure Access**
	GPCRC Secure Access			
29	IFADCDEBUG	0x1	RW	**IFADCDEBUG Secure Access**
	IFADCDEBUG Secure Access			
28	BURAM	0x1	RW	**BURAM Secure Access**
	BURAM Secure Access			
27	SYSCFG	0x1	RW	**SYSCFG Secure Access**
	SYSCFG Secure Access			
26	SYSCFGCFGNS	0x1	RW	**SYSCFGCFGNS Secure Access**
	SYSCFGCFGNS Secure Access			
25	CHIPTESTCTRL	0x1	RW	**CHIPTESTCTRL Secure Access**
	CHIPTESTCTRL Secure Access			
24	I2C1	0x1	RW	**I2C1 Secure Access**
	I2C1 Secure Access			
23	BURTC	0x1	RW	**BURTC Secure Access**
	BURTC Secure Access			
22	USART1	0x1	RW	**USART1 Secure Access**
	USART1 Secure Access			
21	USART0	0x1	RW	**USART0 Secure Access**
	USART0 Secure Access			
20	TIMER4	0x1	RW	**TIMER4 Secure Access**
	TIMER4 Secure Access			
19	TIMER3	0x1	RW	**TIMER3 Secure Access**
	TIMER3 Secure Access			
18	TIMER2	0x1	RW	**TIMER2 Secure Access**

Bit	Name	Reset	Access	Description
	TIMER2 Secure Access			
17	TIMER1	0x1	RW	**TIMER1 Secure Access**
	TIMER1 Secure Access			
16	TIMER0	0x1	RW	**TIMER0 Secure Access**
	TIMER0 Secure Access			
15	LDMAXBAR	0x1	RW	**LDMAXBAR Secure Access**
	LDMAXBAR Secure Access			
14	LDMA	0x1	RW	**LDMA Secure Access**
	LDMA Secure Access			
13	GPIO	0x1	RW	**GPIO Secure Access**
	GPIO Secure Access			
12	PRS	0x1	RW	**PRS Secure Access**
	PRS Secure Access			
11	ICACHE0	0x1	RW	**ICACHE0 Secure Access**
	ICACHE0 Secure Access			
10	MSC	0x1	RW	**MSC Secure Access**
	MSC Secure Access			
9	ULFRCO	0x1	RW	**ULFRCO Secure Access**
	ULFRCO Secure Access			
8	LFRCO	0x1	RW	**LFRCO Secure Access**
	LFRCO Secure Access			
7	LFXO	0x1	RW	**LFXO Secure Access**
	LFXO Secure Access			
6	DPLL0	0x1	RW	**DPLL0 Secure Access**
	DPLL0 Secure Access			
5	FSRCO	0x1	RW	**FSRCO Secure Access**
	FSRCO Secure Access			
4	HFRCO0	0x1	RW	**HFRCO0 Secure Access**
	HFRCO0 Secure Access			
3	HFXO0	0x1	RW	**HFXO0 Secure Access**
	HFXO0 Secure Access			
2	CMU	0x1	RW	**CMU Secure Access**
	CMU Secure Access			
1	EMU	0x1	RW	**EMU Secure Access**
	EMU Secure Access			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.10 SMU_PPUSATD1 - PPU SATD Register 1

Offset	Bit Position																													
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	
Access																	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
Name																	AHBRADIO	CRYPTOACC	EUART0	AMUXCP0	WDOG0	I2C0	IADC0	LETIMER0	RTCC	SMUCFGNS	SMU	RADIOAES	RFSENSE	PDM	DCDC	

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	AHBRADIO	0x1	RW	**AHBRADIO Secure Access**
	AHBRADIO Secure Access			
14	CRYPTOACC	0x1	RW	**CRYPTOACC Secure Access**
	CRYPTOACC Secure Access			
13	EUART0	0x1	RW	**EUART0 Secure Access**
	EUART0 Secure Access			
12	AMUXCP0	0x1	RW	**AMUXCP0 Secure Access**
	AMUXCP0 Secure Access			
11	WDOG0	0x1	RW	**WDOG0 Secure Access**
	WDOG0 Secure Access			
10	I2C0	0x1	RW	**I2C0 Secure Access**
	I2C0 Secure Access			
9	IADC0	0x1	RW	**IADC0 Secure Access**
	IADC0 Secure Access			
8	LETIMER0	0x1	RW	**LETIMER0 Secure Access**
	LETIMER0 Secure Access			
7	RTCC	0x1	RW	**RTCC Secure Access**
	RTCC Secure Access			
6	SMUCFGNS	0x1	RW	**SMUCFGNS Secure Access**
	SMUCFGNS Secure Access			
5	SMU	0x1	RW	**SMU Secure Access**
	SMU Secure Access			
4	RADIOAES	0x1	RW	**RADIOAES Secure Access**
	RADIOAES Secure Access			
3	RFSENSE	0x1	RW	**RFSENSE Secure Access**
	RFSENSE Secure Access			

Bit	Name	Reset	Access	Description
2	PDM PDM Secure Access	0x1	RW	**PDM Secure Access**
1	DCDC DCDC Secure Access	0x1	RW	**DCDC Secure Access**
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.11 SMU_PPUFS - PPU Fault Status

Offset	Bit Position																													
0x140	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												R				
Name																												PPUFSPERIPHID				

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	PPUFSPERIPHID ID of the peripheral that caused the fault.	0x0	R	**Peripheral ID**

10.5.12 SMU_BMPUPATD0 - BMPU PATD Register 0

Offset	Bit Position																													
0x150	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x1	0x1	0x1	0x1	0x1
Access																												RW	RW	RW	RW	RW
Name																												LDMA	RADIOIFADCDEBUG	RADIOSUBSYSTEM	CRYPTOACC	RADIOAES

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	LDMA	0x1	RW	**MCU LDMA privileged mode**
	MCU LDMA privileged mode			
3	RADIOIFADCDEBUG	0x1	RW	**RADIO IFADC debug privileged mode**
	RADIO IFADC debug write privileged mode			
2	RADIOSUBSYSTEM	0x1	RW	**RADIO subsystem masters privileged mode**
	RADIO subsystem masters (FRC and SEQ) privileged mode			
1	CRYPTOACC	0x1	RW	**CRYPTOACC DMA privileged mode**
	CRYPTOACC DMA privileged mode			
0	RADIOAES	0x1	RW	**RADIO AES DMA privileged mode**
	RADIOAES DMA privileged mode			

10.5.13 SMU_BMPUSATD0 - BMPU SATD Register 0

Offset	Bit Position																													
0x170	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x1	0x1	0x1	0x1	0x1
Access																												RW	RW	RW	RW	RW
Name																												LDMA	RADIOIFADCDEBUG	RADIOSUBSYSTEM	CRYPTOACC	RADIOAES

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	LDMA	0x1	RW	**MCU LDMA secure mode**
	MCU LDMA secure mode			
3	RADIOIFADCDEBUG	0x1	RW	**RADIO IFADC debug secure mode**
	RADIO IADC debug write secure mode			
2	RADIOSUBSYSTEM	0x1	RW	**RADIO subsystem masters secure mode**
	RADIO subsystem masters (FRC and SEQ) secure mode			
1	CRYPTOACC	0x1	RW	**CRYPTOACC DMA secure mode**
	CRYPTOACC DMA secure mode			
0	RADIOAES	0x1	RW	**RADIOAES DMA secure mode**
	RADIOAES DMA secure mode			

10.5.14 SMU_BMPUFS - BMPU Fault Status

Offset	Bit Position																													
0x250	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									BMPUFSMASTERID							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	BMPUFSMASTERID	0x0	R	**Master ID**
	ID of master that triggered fault.			

10.5.15 SMU_BMPUFSADDR - BMPU Fault Status Address

Offset	Bit Position																													
0x254	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0																
Access																R																
Name																BMPUFSADDR																

Bit	Name	Reset	Access	Description
31:0	BMPUFSADDR	0x0	R	**Fault Address**
	Access address that triggered fault.			

10.5.16 SMU_ESAURTYPES0 - ESAU Region Types Register 0

Offset	Bit Position																															
0x260	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																				0x0												
Access																				RW												
Name																				ESAUR3NS												

Bit	Name	Reset	Access	Description
31:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	ESAUR3NS	0x0	RW	**Region 3 Non-Secure Type** Set to 1 to configure Region 3 as Non-secure.
11:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.17 SMU_ESAURTYPES1 - ESAU Region Types Register 1

Offset	Bit Position																															
0x264	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																				0x0												
Access																				RW												
Name																				ESAUR11NS												

Bit	Name	Reset	Access	Description
31:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	ESAUR11NS	0x0	RW	**Region 11 Non-Secure Type** Set to 1 to configure Region 11 as Non-secure.
11:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.18 SMU_ESAUMRB01 - ESAU Movable Region Boundary 0-1

Offset	Bit Position																															
0x270	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x2000																											
Access					RW																											
Name					ESAUMRB01																											

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:12	ESAUMRB01	0x2000	RW	**Moveable Region Boundary 0-1** Moveable Region Boundary between Region 0 and Region 1. Address Represents the start of Region 1 at a 4kB offset.
11:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.19 SMU_ESAUMRB12 - ESAU Movable Region Boundary 1-2

Offset	Bit Position																															
0x274	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x4000																											
Access					RW																											
Name					ESAUMRB12																											

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:12	ESAUMRB12	0x4000	RW	**Moveable Region Boundary 1-2** Moveable Region Boundary between Region 1 and Region 2. Address Represents the start of Region 2 at a 4kB offset.
11:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.20 SMU_ESAUMRB45 - ESAU Movable Region Boundary 4-5

Offset	Bit Position																															
0x280	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x2000																											
Access					RW																											
Name					ESAUMRB45																											

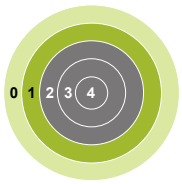
Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:12	ESAUMRB45	0x2000	RW	**Moveable Region Boundary 4-5**
	Moveable Region Boundary between Regions 4 and 5. This represents the starting address of Region 5.			
11:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

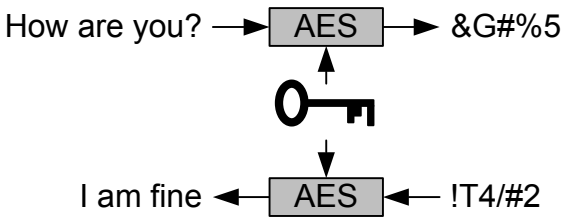
10.5.21 SMU_ESAUMRB56 - ESAU Movable Region Boundary 5-6

Offset	Bit Position																															
0x284	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x4000																											
Access					RW																											
Name					ESAUMRB56																											

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:12	ESAUMRB56	0x4000	RW	**Moveable Region Boundary 5-6**
	Moveable Region Boundary between Regions 5 and 6. This represents the starting address of Region 6.			
11:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

11. CRYPTOACC - Cryptographic Accelerator





Quick Facts

What?

The Cryptographic Accelerator provides fast, energy-efficient hardware for symmetric-key cryptographic operations, a Public Key Engine (PKE) for asymmetric key operations, and non-deterministic random number generation (RNG).

Why?

Efficient hardware-based cryptography helps to meet the speed and energy demands of secure applications while providing additional protection against common attacks.

How?

Dedicated hardware allows fast processing with little to no CPU intervention.

11.1 Introduction

The CRYPTOACC (Cryptographic Accelerator) includes hardware for symmetric-key cryptographic operations such as AES, a Public Key Engine (PKE) for asymmetric key operations, and a non-deterministic random number generator (RNG), suitable for key generation. CRYPTOACC also provides a DMA bus master for efficient data transfer into and out of the block.

11.2 Features

- Acceleration of cryptographic functions
  - AES encryption and decryption with 128, 192, or 256-bit keys
  - SHA-1 and SHA-2 up to 256-bit
  - Supported block cipher modes of operation for AES include: ECB, CTR, CBC, CBC-MAC, CMAC, CCM, and GCM.
  - ECC over GF(P) up to 256-bit
  - Supported ECC NIST recommended curves include P-192 and P-256
- True Random Number Generation
  - Entropy Source complies to NIST 800-90B requirements
  - Online Health tests comply to NIST 800-90B and AIS31 requirements
  - Random Data Passes NIST 800-22 and NIST 800-90B test suites

11.3 Cryptographic Functions

The CRYPTOACC allows efficient acceleration of common cryptographic operations with a low CPU load. CRYPTOACC can implement or accelerate Elliptic Curve Cryptography (ECC), Secure Hash Algorithm (SHA-1, SHA-224, SHA-256), and various block cipher modes based on the Advanced Encryption Standard (AES).

Extensive software support is provided via mbed TLS plugins. Many security systems fail due to mistakes in the implementation. Therefore implementations should be left to experts in cryptographic algorithms. The implemented security functions are described in the mbed TLS documentation located at http://docs.silabs.com.

Control and access registers for the cryptographic acceleration functions are described in the following sections. Although the register interface is provided here, it is highly recommended to use the APIs provided by mbed TLS.

11.3.1 CRYPTOACC_PKCTRL Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	CRYPTOACC_POINTER	RW	Pointers
0x004	CRYPTOACC_COMMAND	RW	Command
0x008	CRYPTOACC_PKCTRL	W	Control
0x00C	CRYPTOACC_PKSTATUS	RH	Status
0x010	CRYPTOACC_VERSION	R	Version
0x014	CRYPTOACC_TIMER	RH	Timer

11.3.2 CRYPTOACC_PKCTRL Register Description

11.3.2.1 CRYPTOACC_POINTER - Pointers

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0								0x0								0x0								0x0			
Access					RW								RW								RW								RW			
Name					OPPTRN								OPPTRC								OPPTRB								OPPTRA			

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:24	OPPTRN	0x0	RW	**OpPtrN**
	When executing primitive arithmetic operations, this pointer defines the location where the modulus is located in memory (location 0 or 0xF).			
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	OPPTRC	0x0	RW	**OpPtrC**
	When executing primitive arithmetic operations, this pointer defines the location (0 to 0xF) where the result will be stored in memory.			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	OPPTRB	0x0	RW	**OpPtrB**
	When executing primitive arithmetic operations, this Pointer defines where operand B is located in memory (location 0 to 0xF).			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	OPPTRA	0x0	RW	**OpPtrA**
	When executing primitive arithmetic operations, this Pointer defines where operand A is located in memory (location 0 to 0xF).			

11.3.2.2 CRYPTOACC_COMMAND - Command

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0	0x0	0x0	0x0				0x0				0x0											0x0	0x0						
Access	RW	RW	RW	RW	RW	RW				RW				RW											RW	RW						
Name	CALCR2	FLAGB	FLAGA	SWAPBYTES	BUFSEL	EDWARDS				SELCURVE				SIZE											FIELD	OPERATION						

Bit	Name	Reset	Access	Description
31	CALCR2	0x0	RW	**Calculate R2**
	This bit indicates if the IP has to calculate R² mod N for the next operation. This bit must be set to '1' when a new prime number has been programmed. This bit is used for primitive operations and ignored for the other operations			
	Value	Mode		Description
	0	FALSE		don't recalculate R² mod N
	1	TRUE		re-calculate R² mod N
30	FLAGB	0x0	RW	**Flag B**
	Flag B			
29	FLAGA	0x0	RW	**Flag A**
	Flag A			
28	SWAPBYTES	0x0	RW	**Swap bytes**
	Swap the bytes on AHB interface. This bit must be programmed before writing/reading any data in data memory. See doc section 4.2.1			
	Value	Mode		Description
	0	NATIVE		Native format (little endian)
	1	SWAPPED		Byte swapped (big endian)
27	BUFSEL	0x0	RW	**Buffer Select**
	Buffer Select			
	Value	Mode		Description
	0	MEM0		use data in data memory 0
26	EDWARDS	0x0	RW	**Edwards Curve Enable**
	Edwards			
25:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	SELCURVE	0x0	RW	**Select Curve**
	Enable accelerator for specific curve modulus. This field has no effect when the optional acceleration hardware is not included.			

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	NONE		No acceleration
	1	P256		P256
	2	P384		P384
	3	P521		P521/E-521
	4	P192		P192
	5	C25519		Curve25519/Ed25519
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:8	SIZE	0x0	RW	**Size of Operands in data memory** This field defines the size (= number of bytes minus one) of the operands for the current operation. Possible values are limited by the maximum supported operand size. See doc for examples.
7	FIELD	0x0	RW	**Field** Field
	Value	Mode		Description
	0	GFP		Field is GF(p)
	1	GF2M		Field is GF(2^m)
6:0	OPERATION	0x0	RW	**Type of Operation** This field defines the operation to be performed. See BA414EP documentation for exhaustive enumeration listing

11.3.2.3 CRYPTOACC_PKCTRL - Control

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															W	W
Name																															IFC	PKSTART

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	IFC	0x0	W	**ClearIRQ** Writing a '1' clears the IRQ output
0	PKSTART	0x0	W	**PK Start** Writing a '1' starts the processing

11.3.2.4 CRYPTOACC_PKSTATUS - Status

Offset	Bit Position																															
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0	0x0			0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0			
Access															R	R			R	R	R	R	R	R	R	R	R	R	R			
Name															PKIF	PKBUSY			NOTQUAD	COMPOSITE	NOTINVERTIBLE	PARAMABNOTVALID	SIGNOTVALID	NOTIMPLEMENTED	PARAMNNOTVALID	COUPLENOTVALID	ATINFINITY	NOTONCURVE	FAILADDR			

Bit	Name	Reset	Access	Description
31:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	PKIF	0x0	R	**Interrupt status**
	This bit reflects the IRQ output value. It is set to '1' when the operation is finished. It is cleared by writing IFC			
16	PKBUSY	0x0	R	**PK busy**
	This bit reflects the BUSY output value. It is set to '1' when the operation starts and it is cleared when the operation is finished			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	NOTQUAD	0x0	R	**Not quadratic residue**
	set to '1' when executing a modular square root if the operand is not a quadratic residue.			
12	COMPOSITE	0x0	R	**Composite**
	updated after the Rabin-Miller Primality test			

Value	Mode	Description
0	FALSE	random number under test is probably prime
1	TRUE	random number under test is composite

11	NOTINVERTIBLE	0x0	R	**Not invertible**
	set to '1' when executing a modular inversion (PK_CommandReg[3:0] = 0x6 or 0x9) if the operand is not invertible			
10	PARAMABNOTVALID	0x0	R	**Param AB not valid**
	set to 1 when parameters A and B are not valid, i.e $4A^3 + 27B^2 = 0$. This flag is updated after execution of the command Check_AB			
9	SIGNOTVALID	0x0	R	**Signature not valid**
	indicates if the signature can be accepted or must be rejected. This flag is set to 1 when the signature is not valid and is updated after execution of the command ECDSA_Generation, ECDSA_Verification, DSA_Generation, DSA_Verification, Ed25519_CheckValid			
8	NOTIMPLEMENTED	0x0	R	**Not implemented**
	set to '1' when the Type of operation programmed in the command register is not supported.			
7	PARAMNNOTVALID	0x0	R	**Param n not valid**

Bit	Name	Reset	Access	Description
	set to 1 when Parameter n is not valid. This flag is updated after execution of the command Check_n.			
6	COUPLENOTVALID	0x0	R	**Couple not valid**
	set to 1 when couple x, y is not valid (i.e. not smaller than the prime). This flag is updated after execution of the command Check_Couple_Less_Prime.			
5	ATINFINITY	0x0	R	**Point Px at infinity**
	set to 1 when Point Px is at the infinity. This flag is updated after execution of an ECC operation			
4	NOTONCURVE	0x0	R	**Point Px not on curve**
	set to 1 when Point Px is not on the defined EC. This flag is updated after execution of the command Check_Point_On-Curve			
3:0	FAILADDR	0x0	R	**Fail Address**
	These bits indicate which data location generated the error flag. They are not available for all error flags.			

11.3.2.5 CRYPTOACC_VERSION - Version

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0								0x0							
Access																	R								R							
Name																	HW								SW							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	HW	0x0	R	**Hardware version number**
	Version of the hardware			
7:0	SW	0x0	R	**Software version number**
	Version of the micro-code			

11.3.2.6 CRYPTOACC_TIMER - Timer

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	TIMER																															

Bit	Name	Reset	Access	Description
31:0	TIMER	0x0	R	**Timer**
	Number of core clock cycles used during previous operation			

11.4 DMA Interface

The CRYPTOACC includes a DMA bus master which allows the block to autonomously fetch data, perform cipher operations and store data across multiple blocks with little CPU intervention.

Control and access registers used for fetching and pushing data with the DMA bus master are documented in the following sections. Although the register interface is provided here, it is highly recommended to use the APIs provided by mbed TLS.

The register map in this section includes several read-only hardware configuration registers. These are used by mbed TLS software to determine device capabilities.

11.4.1 CRYPTOACC Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	CRYPTOACC_FETCHADDR	RW	Fetcher Address
0x008	CRYPTOACC_FETCHLEN	RW	Fetcher Length
0x00C	CRYPTOACC_FETCHTAG	RW	Fetcher Tag
0x010	CRYPTOACC_PUSHADDR	RW	Pusher Address
0x018	CRYPTOACC_PUSHLEN	RW	Pusher Length
0x01C	CRYPTOACC_IEN	RWH	Interrupt Enable
0x028	CRYPTOACC_IF	RH INTFLAG	Interrupt Flags
0x030	CRYPTOACC_IF_CLR	WH	Interrupt status clear
0x034	CRYPTOACC_CTRL	RW	Control register
0x038	CRYPTOACC_CMD	W	Command register
0x03C	CRYPTOACC_STATUS	RH	Status register
0x400	CRYP-TOACC_INCL_IPS_HW_CFG	RH	General CRYPTOACC Hardware Configuration
0x404	CRYP-TOACC_BA411E_HW_CFG_1	RH	BA411E Hardware Configuration 1
0x408	CRYP-TOACC_BA411E_HW_CFG_2	RH	BA411E Hardware Configuration 2
0x40C	CRYPTOACC_BA413_HW_CFG	RH	BA413 Hardware Configuration
0x410	CRYPTOACC_BA418_HW_CFG	RH	BA418 Hardware Configuration
0x414	CRYPTOACC_BA419_HW_CFG	RH	BA419 Hardware Configuration

11.4.2 CRYPTOACC Register Description

11.4.2.1 CRYPTOACC_FETCHADDR - Fetcher Address

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	ADDR																															

Bit	Name	Reset	Access	Description
31:0	ADDR	0x0	RW	**Start address of data block**
	Fetch address			

11.4.2.2 CRYPTOACC_FETCHLEN - Fetcher Length

Offset	Bit Position																															
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x0	0x0	0x0																											
Access			RW	RW	RW																											
Name			REALIGN	CONSTADDR	LENGTH																											

Bit	Name	Reset	Access	Description
31:30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	REALIGN	0x0	RW	**Realign length**
	Fetch realign length			
28	CONSTADDR	0x0	RW	**Constant address**
	Fetch constant address			
27:0	LENGTH	0x0	RW	**Length of data block**
	Fetch length			

11.4.2.3 CRYPTOACC_FETCHTAG - Fetcher Tag

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	TAG																															

Bit	Name	Reset	Access	Description
31:0	TAG	0x0	RW	**User tag**
	User tag			

11.4.2.4 CRYPTOACC_PUSHADDR - Pusher Address

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	ADDR																															

Bit	Name	Reset	Access	Description
31:0	ADDR	0x0	RW	**Start address of data block**
	Push starting address			

11.4.2.5 CRYPTOACC_PUSHLEN - Pusher Length

Offset	Bit Position																															
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0	0x0	0x0	0x0																											
Access		RW	RW	RW	RW																											
Name		DISCARD	REALIGN	CONSTADDR	LENGTH																											

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30	DISCARD	0x0	RW	**Discard data**
	Discard data			
29	REALIGN	0x0	RW	**Realign length**
	Realign length			
28	CONSTADDR	0x0	RW	**Constant address**
	Constant address			
27:0	LENGTH	0x0	RW	**Start address of data block**
	Starting address of data block			

11.4.2.6 CRYPTOACC_IEN - Interrupt Enable

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											RW	RW	RW	RW	RW	RW
Name																											PUSHERERROR	PUSHERSTOPPED	PUSHERENDOFBLOCK	FETCHERERROR	FETCHERSTOPPED	FETCHERENDOFBLOCK

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	PUSHERERROR Error interrupt enable	0x0	RW	**Error interrupt enable**
4	PUSHERSTOPPED Stopped interrupt enable	0x0	RW	**Stopped interrupt enable**
3	PUSHERENDOF-BLOCK End of block interrupt enable	0x0	RW	**End of block interrupt enable**
2	FETCHERERROR Error interrupt enable	0x0	RW	**Error interrupt enable**
1	FETCHERSTOPPED Stopped interrupt enable	0x0	RW	**Stopped interrupt enable**
0	FETCHERENDOF-BLOCK End of block interrupt enable	0x0	RW	**End of block interrupt enable**

11.4.2.7 CRYPTOACC_IF - Interrupt Flags

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											R	R	R	R	R	R
Name																											PUSHERERROR	PUSHERSTOPPED	PUSHERENDOFBLOCK	FETCHERERROR	FETCHERSTOPPED	FETCHERENDOFBLOCK

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	PUSHERERROR	0x0	R	**Error interrupt flag**
	Triggered when an error response is received from AXI			
4	PUSHERSTOPPED	0x0	R	**Stopped interrupt flag**
	Triggered when reaching a block with Stop=1 (or end of direct transfer)			
3	PUSHERENDOF-BLOCK	0x0	R	**End of block interrupt flag**
	Triggered at the end of each block (if enabled in the descriptor - scatter-gather only)			
2	FETCHERERROR	0x0	R	**Error interrupt flag**
	Triggered when an error response is received from AXI			
1	FETCHERSTOPPED	0x0	R	**Stopped interrupt flag**
	Triggered when reaching a block with Stop=1 (or end of direct transfer)			
0	FETCHERENDOF-BLOCK	0x0	R	**End of block interrupt flag**
	Triggered at the end of each block (if enabled in the descriptor - scatter-gather only)			

11.4.2.8 CRYPTOACC_IF_CLR - Interrupt status clear

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											W	W	W	W	W	W
Name																											PUSHERERROR	PUSHERSTOPPED	PUSHERENDOFBLOCK	FETCHERERROR	FETCHERSTOPPED	FETCHERENDOFBLOCK

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	PUSHERERROR	0x0	W	**Error interrupt flag clear**
	Error interrupt flag clear			
4	PUSHERSTOPPED	0x0	W	**Stopped interrupt flag clear**
	Stopped interrupt flag clear			
3	PUSHERENDOF-BLOCK	0x0	W	**End of block interrupt flag clear**
	End of block interrupt flag clear			
2	FETCHERERROR	0x0	W	**Error interrupt flag clear**
	Error interrupt flag clear			
1	FETCHERSTOPPED	0x0	W	**Stopped interrupt flag clear**
	Stopped interrupt flag clear			
0	FETCHERENDOF-BLOCK	0x0	W	**End of block interrupt flag clear**
	End of block interrupt flag clear			

11.4.2.9 CRYPTOACC_CTRL - Control register

Offset	Bit Position																													
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												RW	RW	RW	RW	RW
Name																												SWRESET	STOPPUSHER	STOPFETCHER	PUSHERSCATTERGATHER	FETCHERSCATTERGATHER

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	SWRESET	0x0	RW	**Software reset**
	When this bit is high, the software reset of the DMA modules, the FIFO's and the processing module will be activated. The AXI bus is not affected (pending transfers will be completed).			
3	STOPPUSHER	0x0	RW	**Stop pusher**
	When this bit is high, the pusher will stop at the end of the current block (even if the STOP bit in the descriptor is low).			
2	STOPFETCHER	0x0	RW	**Stop fetcher**
	When this bit is high, the fetcher will stop at the end of the current block (even if the STOP bit in the descriptor is low).			
1	PUSHERSCATTER-GATHER	0x0	RW	**Pusher scatter/gather**
	When this bit is zero, the pusher runs in direct mode. When this bit is one, the pusher runs in scatter-gather mode.			
0	FETCHERSCATTER-GATHER	0x0	RW	**Fetcher scatter/gather**
	When this bit is zero, the fetcher runs in direct mode. When this bit is one, the fetcher runs in scatter-gather mode.			

11.4.2.10 CRYPTOACC_CMD - Command register

Offset	Bit Position																													
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															W	W
Name																															STARTPUSHER	STARTFETCHER

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	STARTPUSHER	0x0	W	**Start push**
	Writing a '1' starts the pusher. Writing a '0' has no effect.			
0	STARTFETCHER	0x0	W	**Start fetch**
	Writing a '1' starts the fetcher. Writing a '0' has no effect.			

11.4.2.11 CRYPTOACC_STATUS - Status register

Offset	Bit Position																													
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																									0x0	0x0	0x0			0x0	0x0
Access	R																									R	R	R			R	R
Name	FIFODATANUM																									SOFTRSTBSY	WAITING	NOTEMPTY			PUSHERBSY	FETCHERBSY

Bit	Name	Reset	Access	Description
31:16	FIFODATANUM	0x0	R	**Number of data in output FIFO**
	Number of bytes in output FIFO (pusher).			
15:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	SOFTRSTBSY	0x0	R	**Software reset busy**
	This bit is high when the soft reset is on-going.			
5	WAITING	0x0	R	**Pusher waiting for FIFO**
	This bit is high when the pusher if waiting for more data in output FIFO.			
4	NOTEMPTY	0x0	R	**Not empty flag from input FIFO (fetcher)**
	Not empty flag from input FIFO (fetcher).			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	PUSHERBSY	0x0	R	**Pusher busy**
	This bit is high as long as the pusher is busy.			
0	FETCHERBSY	0x0	R	**Fetcher busy**
	This bit is high as long as the fetcher is busy.			

11.4.2.12 CRYPTOACC_INCL_IPS_HW_CFG - General CRYPTOACC Hardware Configuration

Offset	Bit Position																												
0x400	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x1	0x1	0x0	0x0	0x0	0x0	0x1	0x0	0x0	0x0	0x1
Access																						R	R	R	R	R	R	R	R	R	R	R
Name																						g_IncludeNDRNG	g_IncludePKE	g_IncludeSM4	g_IncludeZUC	g_IncludeSHA3	g_IncludeChachaPoly	g_IncludeHASH	g_IncludeDES	g_IncludeAESXTS	g_IncludeAESGCM	g_IncludeAES

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	g_IncludeNDRNG	0x1	R	**Generic g_IncludeNDRNG value**
	BA431 – NDRNG IP included if set			
9	g_IncludePKE	0x1	R	**Generic g_IncludePKE value**
	BA414EP – PKE IP included if set			
8	g_IncludeSM4	0x0	R	**Generic g_IncludeSM4 value**
	BA419 – SM4 IP included if set			
7	g_IncludeZUC	0x0	R	**Generic g_IncludeZUC value**
	BA421 – ZUC IP included if set			
6	g_IncludeSHA3	0x0	R	**Generic g_IncludeSHA3 value**
	BA418 – SHA3 IP included if set			
5	g_IncludeChachaPoly	0x0	R	**Generic g_IncludeChachaPoly value**
	BA417 – ChachaPoly IP included if set			
4	g_IncludeHASH	0x1	R	**Generic g_IncludeHASH value**
	BA413 – HASH IP included if set			
3	g_IncludeDES	0x0	R	**Generic g_IncludeDES value**
	BA412 – DES IP included if set			
2	g_IncludeAESXTS	0x0	R	**Generic g_IncludeAESXTS value**
	BA416 – AESXTS IP included if set			
1	g_IncludeAESGCM	0x0	R	**Generic g_IncludeAESGCM value**
	BA415 – AESGCM IP included if set			
0	g_IncludeAES	0x1	R	**Generic g_IncludeAES value**
	BA411E – AES IP included if set			

11.4.2.13 CRYPTOACC_BA411E_HW_CFG_1 - BA411E Hardware Configuration 1

Offset	Bit Position																															
0x404	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset						0x7									0x0	0x0								0x17F								
Access						R									R	R								R								
Name						g_Keysize									g_UseMasking	g_CS								g_AesModesPoss								

Bit	Name	Reset	Access	Description
31:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:24	g_Keysize	0x7	R	**Generic g_Keysize value**
	BA411E AES engine configuration			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	g_UseMasking	0x0	R	**Generic g_UseMasking value**
	BA411E AES engine configuration			
16	g_CS	0x0	R	**Generic g_CS value**
	BA411E AES engine configuration			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	g_AesModesPoss	0x17F	R	**AES Modes Supported**
	BA411E AES engine configuration			

11.4.2.14 CRYPTOACC_BA411E_HW_CFG_2 - BA411E Hardware Configuration 2

Offset	Bit Position																															
0x408	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x80															
Access																	R															
Name																	g_CtrSize															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	g_CtrSize BA411E AES engine configuration	0x80	R	**Generic g_CtrSize value**

11.4.2.15 CRYPTOACC_BA413_HW_CFG - BA413 Hardware Configuration

Offset	Bit Position																															
0x40C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0x0	0x1	0x1													0x7F			
Access														R	R	R													R			
Name														g_HashVerifyDigest	g_HMAC_enabled	g_HashPadding													g_HashMaskFunc			

Bit	Name	Reset	Access	Description
31:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	g_HashVerifyDigest BA413 Hash engine configuration	0x0	R	**Generic g_HashVerifyDigest value**
17	g_HMAC_enabled BA413 Hash engine configuration	0x1	R	**Generic g_HMAC_enabled value**
16	g_HashPadding BA413 Hash engine configuration	0x1	R	**Generic g_HashPadding value**
15:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	g_HashMaskFunc BA413 Hash engine configuration	0x7F	R	**Generic g_HashMaskFunc value**

11.4.2.16 CRYPTOACC_BA418_HW_CFG - BA418 Hardware Configuration

Offset	Bit Position																														
0x410	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x1
Access																																R
Name																																g_Sha3CtxtEn

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	g_Sha3CtxtEn BA418 SHA3 configuration	0x1	R	**Generic g_Sha3CtxtEn value**

11.4.2.17 CRYPTOACC_BA419_HW_CFG - BA419 Hardware Configuration

Offset	Bit Position																														
0x414	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x5F			
Access																													R			
Name																													g_SM4ModesPoss			

Bit	Name	Reset	Access	Description
31:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	g_SM4ModesPoss BA419 SM4 engine configuration	0x5F	R	**Generic g_SM4ModesPoss value**

11.5 Random Number Generation

The CRYPTOACC provides access to a non-deterministic random number generator based on a full hardware solution. The RNG output passes the NIST 800-22 and AIS31 test suites. The RNG module includes several built-in self tests to detect issues with the noise source, ensure entropy, and meet cryptography standards. The Repetition Count Test and Adaptive Proportion Test with window sizes of 64 and 4096 bits described in section 6.5.1.2 of NIST-800-90B are implemented in hardware and run continuously on the data.

http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf

The AIS31 Online Test described in section 5.5.3 of AIS 31 is also implemented in hardware, and runs continuously on the data.

https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf

Control and access registers for the random number generator are described in the following sections. Although the register interface is provided here, it is highly recommended to use the APIs provided by mbed TLS.

11.5.1 CRYPTOACC_RNGCTRL Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	CRYPTOACC_RNGCTRL	RWH	RNG Control Register
0x004	CRYPTOACC_FIFOLEVEL	RH	FIFO Level Register
0x008	CRYPTOACC_FIFOTHRESH	RH	FIFO Threshold Register
0x00C	CRYPTOACC_FIFODEPTH	R	FIFO Depth Register
0x010	CRYPTOACC_KEYx	RW	Key Register
0x020	CRYPTOACC_TESTDATA	W	Test Data Register
0x030	CRYPTOACC_RNGSTATUS	RWH	RNG Status Register
0x034	CRYPTOACC_INITWAITVAL	RW	Initial Wait Counter
0x040	CRYPTOACC_SWOFFTMRVAL	RW	Switch off timer value
0x044	CRYPTOACC_CLKDIV	RW	Sample clock divider
0x048	CRYPTOACC_AIS31CONF0	RW	AIS31 configuration 0 register
0x04C	CRYPTOACC_AIS31CONF1	RW	AIS31 configuration 1 register
0x050	CRYPTOACC_AIS31CONF2	RW	AIS31 configuration 2 register
0x054	CRYPTOACC_AIS31STATUS	RWH	AIS31 status register

11.5.2 CRYPTOACC_RNGCTRL Register Description

11.5.2.1 CRYPTOACC_RNGCTRL - RNG Control Register

Offset	Bit Position																														
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x0	0x4				0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0		0x0
Access												RW	RW				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW
Name												FIFOWRSTARTUP	NB128BITBLOCKS				AIS31TESTSEL	HEALTHTESTSEL	BYPAIS31	BYPNIST	FORCERUN	ALMIEN	PREIEN	SOFTRESET	FULLIEN	APT4096IEN	APT64IEN	REPCOUNTIEN	CONDBYPASS	TESTEN		ENABLE

Bit	Name	Reset	Access	Description
31:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	FIFOWRSTARTUP	0x0	RW	**Fifo Write Start Up**
	Fifo Write Start Up			
19:16	NB128BITBLOCKS	0x4	RW	**Number of 128b blocks in AES-CBCMAC**
	Number of 128 bit blocks used in AES-CBCMAC post-processing. This value cannot be zero			
15	AIS31TESTSEL	0x0	RW	**AIS31 test input select**
	AIS31 test input select			
	Value	Mode		Description
	0	BEFORE		Before conditioning
	1	AFTER		After conditioning
14	HEALTHTESTSEL	0x0	RW	**Health test input select**
	Health test input select			
	Value	Mode		Description
	0	BEFORE		Before conditioning
	1	AFTER		After conditioning
13	BYPAIS31	0x0	RW	**AIS31 Start-up Test Bypass.**
	Bypass for AIS31 startup test.			
	Value	Mode		Description
	0	NORMAL		AIS31 startup test is applied. No data will be written to the FIFO until the test passes.
	1	BYPASS		AIS31 startup test is bypassed.
12	BYPNIST	0x0	RW	**NIST Start-up Test Bypass.**

Bit	Name	Reset	Access	Description
	Bypass for NIST-800-90B startup test.			
	Value	Mode		Description
	0	NORMAL		NIST-800-90B startup test is applied. No data will be written to the FIFO until the test passes.
	1	BYPASS		NIST-800-90B startup test is bypassed.
11	FORCERUN	0x0	RW	**Oscillator Force Run**
	Set this bit to force oscillators to run even when FIFO is full.			
	Value	Mode		Description
	0	NORMAL		Oscillators will shut down when FIFO is full
	1	RUN		Oscillators will continue to run even after FIFO is full
10	ALMIEN	0x0	RW	**IRQ enable for AIS31 noise alarm**
	Enable/disable AIS31 noise alarm interrupt.			
9	PREIEN	0x0	RW	**IRQ enable for AIS31 prelim. noise alarm**
	Enable/disable AIS31 preliminary noise alarm interrupt.			
8	SOFTRESET	0x0	RW	**Software Reset**
	Set to reset the module. This bit is not cleared automatically.			
	Value	Mode		Description
	0	NORMAL		Module not in reset
	1	RESET		The continuous test, the conditioning function and the FIFO are reset
7	FULLIEN	0x0	RW	**IRQ enable for FIFO full**
	Enable/Disable FIFO full interrupt.			
6	APT4096IEN	0x0	RW	**IRQ enable for APT4096IF**
	Enable/Disable 4096-sample Adaptive Proportion test failure interrupt.			
5	APT64IEN	0x0	RW	**IRQ enable for APT64IF**
	Enable/Disable 64-sample Adaptive Proportion test failure interrupt.			
4	REPCOUNTIEN	0x0	RW	**IRQ enable for Repetition Count Test**
	Enable/Disable Repetition Count Test failure interrupt.			
3	CONDBYPASS	0x0	RW	**Conditioning Bypass**
	Enables bypassing of the conditioning function (to observe entropy source directly).			
	Value	Mode		Description
	0	NORMAL		The conditionig function is used
	1	BYPASS		The conditioning function is bypassed
2	TESTEN	0x0	RW	**Test Enable**
	Selects the input for the conditioning function and continuous tests.			

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	NOISE		Non-determinsitc random number generation
	1	TESTDATA		Pseudo-random number generation
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	ENABLE	0x0	RW	**TRNG Module Enable**
	Enable the TRNG. The module will generate random numbers unless the FIFO is full.			
	Value	Mode		Description
	0	DISABLED		Module disabled
	1	ENABLED		Module enabled

11.5.2.2 CRYPTOACC_FIFOLEVEL - FIFO Level Register

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	FIFOLEVEL																															

Bit	Name	Reset	Access	Description
31:0	FIFOLEVEL	0x0	R	**FIFO Level**
	Number of 32-bit words of random data available in the FIFO.			

11.5.2.3 CRYPTOACC_FIFOTHRESH - FIFO Threshold Register

Offset	Bit Position																															
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x3F																															
Access	R																															
Name	FIFOTHRESH																															

Bit	Name	Reset	Access	Description
31:0	FIFOTHRESH	0x3F	R	**FIFO threshold level**
	FIFO level at which the rings are restarted when in the FIFOFull_Off state, expressed in number of 128bit blocks.			

11.5.2.4 CRYPTOACC_FIFODEPTH - FIFO Depth Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x40																															
Access	R																															
Name	FIFODEPTH																															

Bit	Name	Reset	Access	Description
31:0	FIFODEPTH	0x40	R	**FIFO Depth.**
	Maximum number of 32-bit words that can be stored in the FIFO.			

11.5.2.5 CRYPTOACC_KEYx - Key Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	KEY																															

Bit	Name	Reset	Access	Description
31:0	KEY	0x0	RW	**Key**
	KEY0, KEY1, KEY2, and KEY3 form the 128-bit AES key used in the conditioning function. KEY0 is the MSB and KEY3 is the LSB.			

11.5.2.6 CRYPTOACC_TESTDATA - Test Data Register

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	W																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x0	W	**Test data input to conditioning tests**
	Test data input to conditioning function or to the continuous tests. Each word written to this register represents 32 bits of input data for the selected test in test mode (CONTROL_TESTEN = 1). TESTDATABUSY in the STATUS register will be set to 1 each time data is written, and will clear to 0 when the next data word can be written. Writes to this register are ignored if the TESTEN bit in the CONTROL register is 0.			

11.5.2.7 CRYPTOACC_RNGSTATUS - RNG Status Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0	0x0	0x0	0x0	0x0	0x0	0x0			0x0
Access																							R	RW	R	R	R	R	R			R
Name																							ALMIF	PREIF	FULLIF	APT4096IF	APT64IF	REPCOUNTIF	STATE			TESTDATABUSY

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	ALMIF	0x0	R	**AIS31 Noise Alarm interrupt status**
	Set when a noise alarm is detected in the AIS31 online test.			
8	PREIF	0x0	RW	**AIS31 Preliminary Noise Alarm IF**
	Set when a preliminary noise alarm is detected in the AIS31 online test.			
7	FULLIF	0x0	R	**FIFO full interrupt status**
	Set when the FIFO is full.			
6	APT4096IF	0x0	R	**4096-sample window Adaptive Prop. IF**
	Set when an Adaptive Proportion test (4096-sample window) failure occurs.			
5	APT64IF	0x0	R	**64-sample window Adaptive Proportion IF**
	Set when an Adaptive Proportion test (64-sample window) failure occurs.			
4	REPCOUNTIF	0x0	R	**Repetition Count Test interrupt status**
	Set when a Repetition Count Test failure occurs.			
3:1	STATE	0x0	R	**State of the control FSM**
	State of the control FSM			

Value	Mode	Description
0	RESET	RESET State
1	STARTUP	STARTUP State
2	FIFOFULLON	FIFOFULLON State
3	FIFOFULLOFF	FIFOFULLOFF State
4	RUNNING	RUNNING State
5	ERROR	ERROR State
6	UNUSED_6	UNUSED
7	UNUSED_7	UNUSED

0	TESTDATABUSY	0x0	R	**Test Data Busy**

Bit	Name	Reset	Access	Description
	Indicates that data written to TESTDATA is being processed.			

Value	Mode	Description
0	IDLE	TESTDATA write is finished processing or no test in progress.
1	BUSY	TESTDATA write is still being processed.

11.5.2.8 CRYPTOACC_INITWAITVAL - Initial Wait Counter

Offset	Bit Position																													
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0xFFFF															
Access																	RW															
Name																	INITWAITVAL															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	INITWAITVAL	0xFFFF	RW	**Wait counter value**
	Number of clock cycles to wait before sampling data from the noise source.			

11.5.2.9 CRYPTOACC_SWOFFTMRVAL - Switch off timer value

Offset	Bit Position																													
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0xFFFF															
Access																	RW															
Name																	SWOFFTMRVAL															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	SWOFFTMRVAL	0xFFFF	RW	**Switch Off Timer Value**
	Number of clock cycles to wait before stopping the rings after the FIFO is full.			

11.5.2.10 CRYPTOACC_CLKDIV - Sample clock divider

Offset	Bit Position																													
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									RW							
Name																									VALUE							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	VALUE	0x0	RW	**Sample clock divider**
	The frequency at which the outputs of the rings are sampled is given by Fs = Fpclk/(CLKDIV + 1).			

11.5.2.11 CRYPTOACC_AIS31CONF0 - AIS31 configuration 0 register

Offset	Bit Position																													
0x048	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x4340																0x1040														
Access		RW																RW														
Name		ONLINETHRESH																STARTUPTHRES														

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:16	ONLINETHRESH Online Threshold	0x4340	RW	**Online Threshold**
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:0	STARTUPTHRES Start-up Threshold	0x1040	RW	**Start-up Threshold**

11.5.2.12 CRYPTOACC_AIS31CONF1 - AIS31 configuration 1 register

Offset	Bit Position																													
0x04C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x3C0																0x680														
Access		RW																RW														
Name		ONLINEREPTHRESH																HEXPECTEDVALUE														

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:16	ONLINEREPTHRESH Online Rep Threshold	0x3C0	RW	**Online Repeat Threshold**
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:0	HEXPECTEDVALUE HExpected Value	0x680	RW	**Expected History Value**

11.5.2.13 CRYPTOACC_AIS31CONF2 - AIS31 configuration 2 register

Offset	Bit Position																													
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x440																0x340														
Access		RW																RW														
Name		HMAX																HMIN														

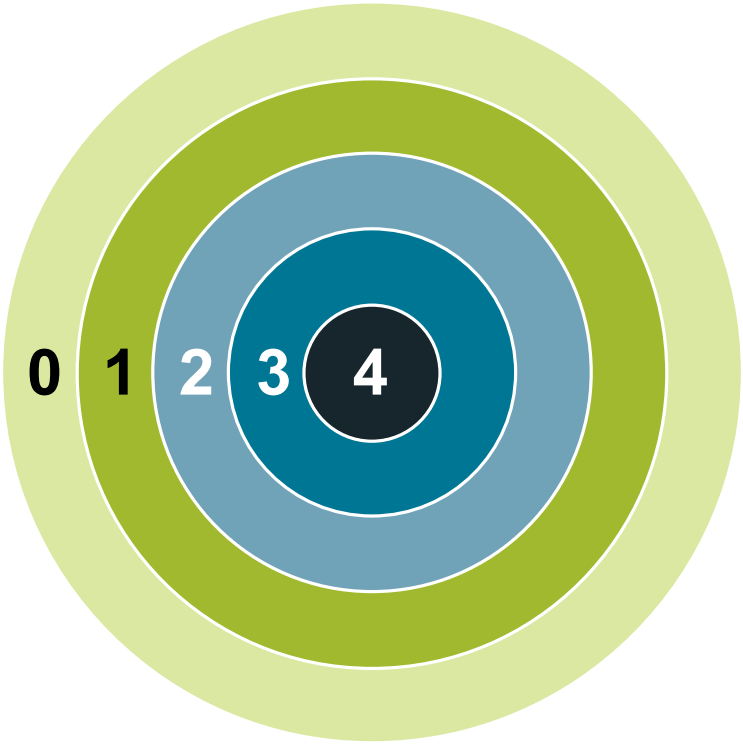
Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:16	HMAX HMax	0x440	RW	**Maximum Allowed History Value**
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:0	HMIN HMin	0x340	RW	**Minimum Allowed History Value**

11.5.2.14 CRYPTOACC_AIS31STATUS - AIS31 status register

Offset	Bit Position																															
0x054	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0	0x0	0x0															
Access															RW	RW	RW															
Name															PRELIMNOISEALARMREP	PRELIMNOISEALARMRNG	NUMPRELIMALARMS															

Bit	Name	Reset	Access	Description
31:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	PRELIMNOISEALARM-REP	0x0	RW	**Preliminary noise alarm Rep**
	Last preliminary noise alarm occurred due to consecutive high X^2			
16	PRELIMNOISEA-LARMRNG	0x0	RW	**Preliminary noise alarm RNG**
	Last preliminary noise alarm occurred due to history value out of range			
15:0	NUMPRELIMALARMS	0x0	RW	**Number of preliminary alarms**
	Number of preliminary noise alarms since counter was last cleared			

12. EMU - Energy Management Unit



Quick Facts

What?

The EMU (Energy Management Unit) handles the different low energy modes in EFR32xG22

Why?

The need for performance and peripheral functions varies over time in most applications. By efficiently scaling the available resources in real time to match the demands of the application, the energy consumption can be kept at a minimum.

How?

With a broad selection of energy modes, a high number of low-energy peripherals available even in EM2, and short wake-up time, applications can dynamically minimize energy consumption during program execution.

12.1 Introduction

The Energy Management Unit (EMU) manages all the low energy modes (EM) in EFR32xG22. Each energy mode manages whether the CPU and the various peripherals are available. The energy modes range from EM0 to EM4. EM0 mode provides the highest amount of features, enabling the CPU, Radio, and peripherals with the highest clock frequency. EM4 Mode provides the lowest power state, allowing the part to return to EM0 on a wake-up condition. The EMU also controls the internal regulators settings and voltage monitoring needed for optimal power configuration and protection.

12.2 Features

The primary features of the EMU are listed below:

- Energy Modes control
  - Entry into EM4
  - Configuration of regulators and clocks for each Energy Mode
  - Configuration of various EM4 wake-up conditions
  - Configuration of GPIO retention settings
- Power routing configurations
  - DCDC control and bypass
- Temperature sensor
- Brown Out Detection
- Supply voltage scaling
  - EM0 / EM1 voltage scaling
  - EM2 / EM3 voltage scaling
- Reset Management
  - Power-on Reset (POR)
  - Brown-out Detection (BOD) on the following power domains:
    - Analog Unregulated Power Domain AVDD
    - Digital Unregulated Power Domain DVDD
    - I/O Unregulated Power Domain IOVDDx
    - Regulated Digital Domain DECOUPLE (DEC)
  - RESETn pin reset
  - Watchdog (WDOG) reset
  - Software triggered reset (SYSRESETREQ)
  - Core LOCKUP condition
  - EM4 Detection
  - EM4 wakeup reset from GPIO pin
  - Configurable reset levels
  - A software readable register indicates the cause of the last reset

12.3 Functional Description

The EMU is responsible for managing the wide range of energy modes available in EFR32xG22. The block works in harmony with the entire platform to easily transition between energy modes in the most efficient manner possible. The following diagram Figure 12.1 EMU Overview on page 287, shows the relative connectivity to the various blocks in the system.

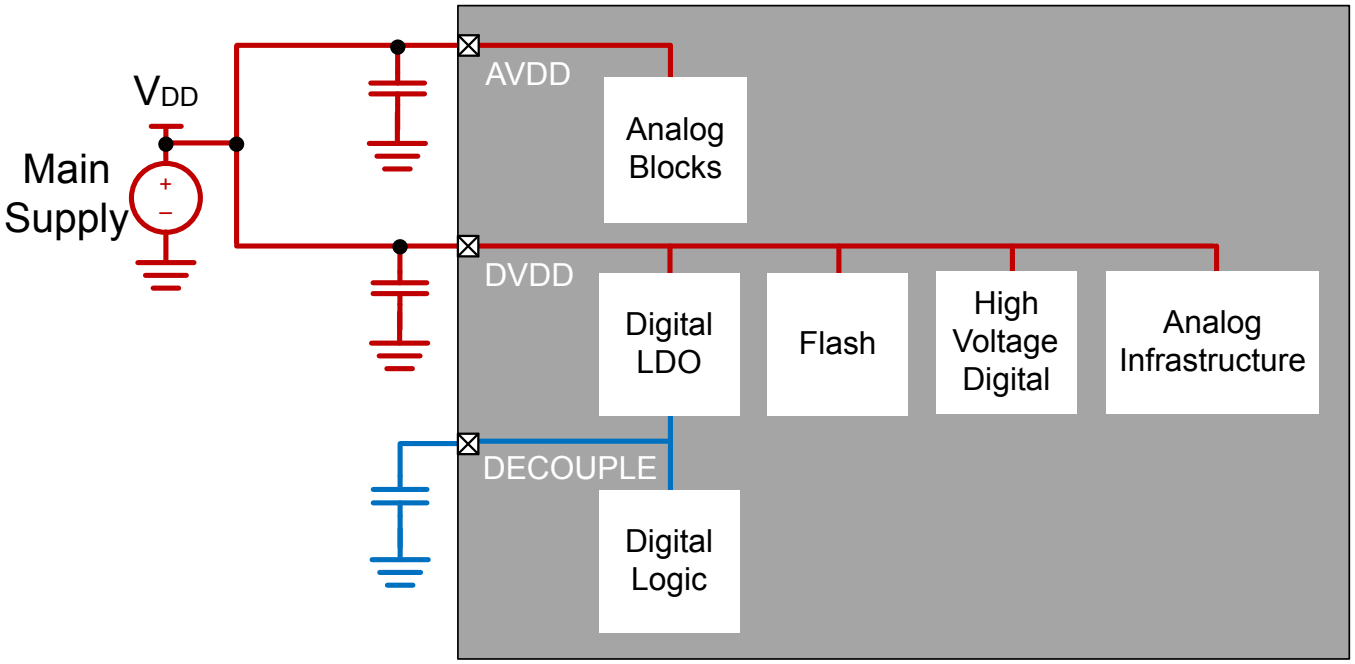


Figure 12.1. EMU Overview

The EMU is available on the peripheral bus. The energy management state machine controls the internal voltage regulators, oscillators, memories, and interrupt system. Events, interrupts, and resets can trigger the energy management state machine to return to the active state. This is further described in the following sections.

12.3.1 Energy Modes

EFR32xG22 features five main energy modes, referred to as Energy Mode 0 (EM0) through Energy Mode 4 (EM4). The Cortex-M33 is only available for program execution in EM0. In EM0 Active/EM1 Sleep any peripheral function can be enabled. EM2 through EM4, also referred to as low energy modes, provide a significantly reduced energy consumption while still allowing a rich set of peripheral functionality. The following Table 12.1 table on page 288 shows the possible transitions between different energy modes.

Table 12.1. Energy Mode Transitions

Current Mode	EM Transition Action				
	Enter EM0	Enter EM1	Enter EM2	Enter EM3	Enter EM4
EM0		Sleep (WFI, WFE)	Deep Sleep (WFI, WFE)	Deep Sleep (WFI, WFE)	EM4 Entry
EM1	IRQ		Peripheral wake up done[1]	Peripheral wake up done [1]	
EM2	IRQ	Peripheral wake up req [1]			
EM3	IRQ	Peripheral wake up req [1]			
EM4	Wake Up				

Note:
1. Peripheral wake-up from EM2/3 to EM1 and then automatically back to EM2/3 when done.

Peripherals such as the IADC and radio have the ability to temporarily wake up the part from either EM2 or EM3 to EM1 in order to transfer data. Once completed, the part is automatically placed back into the EM2 or EM3 mode.

The Core can always request to go to EM1 with the WFI or WFE command during EM0. The core will be prevented from entering EM2 or EM3 if the radio is transferring data or if flash is programming or erasing.

An overview of supported energy modes and available functionality is shown in the following table. For each energy mode, the system will typically default to its lowest power configuration, with non-essential clocks and peripherals disabled. Functionality may be then selectively enabled by software.

Table 12.2. Energy Modes

	EM0 / EM1	EM2	EM3	EM4
Wake-up time to EM0 / EM1	-	TBD [1]	TBD [1]	TBD [1]
Cortex-M33 Core Active	Yes, in EM0 only	-	-	-
Debug	Available	See Note [2]	See Note [2]	-
Digital logic and system RAM retained	Yes	Yes	Yes	-
Flash Memory Access	Available	-	-	-
LDMA (Linked DMA Controller)	Available	Available [3]	Available [3]	-
Radio Controller	Available	Available [4]	-	-
RFSENSE	Available	Available	Available	Available

	EM0 / EM1	EM2	EM3	EM4
High Frequency Oscillators (HFXO,HFRCODPLL) and Clocks (BUSCLK, HCLK, PCLK, RADIOCLK, EM01GRPACLK)	Available	-	-	-
Fast Startup RC Oscillator (FSRCO) and ADC Clock (IADCCLK)	Available	Available [5]	Available [5]	-
Low Frequency Oscillators (LFRCO, LFXO)	Available	Available	-	Available [6]
Low Energy Clocks (EM23GRPACLK, WDOGCLK, RTCCCLK, PRORTCCLK)	Available	Available	Available [7]	-
EM4 Clock (EM4GRPACLK)	Available	Available	Available [7]	Available [6]
ULFRCO (Ultra Low Frequency Oscillator)	On	On	On	Available[6]
CRYPTOACC (Crypto Accelerator)	Available	-	-	-
GPCRC (General Purpose Cyclic Redundancy Check)	Available	-	-	-
BURTC	Available	Available	Available [7]	Available
RTCC	Available	Available	Available[7]	-
RTCC Memory Retained	Yes	Yes	Yes	-
USART (UART/SPI)	Available	-	-	-
EUART	Available	Available[11]	-	-
I^2C	Available	Available [8 11]	Available [8]	-
TIMER (Timer/Counter)	Available	-	-	-
LETIMER (Low Energy Timer)	Available	Available[11]	Available [7 11]	-
WDOG (Watchdog)	Available	Available [11]	Available [7 11]	-
IADC (Analog to Digital Converter)	Available	Available [3 11]	Available [3 11]	-
PDM	Available	-	-	-
DC-DC	Available	Available	Available	-
EMU Temperature Change	Available	Available	Available	-
Brown-Out Detect/Power-on Reset	Available	Available	Available	Available
Pin Reset	Available	Available	Available	Available
PRS (Peripheral Reflex System)	Available	Available[11]	Available[11]	-
GPIO Pin Interrupts	Available	Available	Available	Available [9]
GPIO Pin State Retention	Yes	Yes	Yes	Available [10]

	EM0 / EM1	EM2	EM3	EM4

Note:

1. Approximate time. Additional 32 us / 100 mV required to scale up from a lower VSCALE setting. Refer to the data sheet for specific timing.
2. Leaving the debugger connected when in EM2 or EM3 will cause the system to enter a higher power EM2 mode in which the high frequency clocks are still enabled and certain core functionality is still powered-up in order to maintain debug-functionality.
3. The LDMA can be used with some low power peripherals (e.g., IADC) in EM2/3. Features required by the LDMA which are not supported in EM2/3 (e.g., HCLK), will be automatically enabled prior to the LDMA transfer and then automatically disabled afterwards.
4. The RAC can be woken via a PRS interrupt to EM1 to transfer data. Once complete, the system will return to EM2.
5. Default off, but kept active if used by the IADC.
6. Default off, but kept active if used by the BURTC
7. Must be using ULFRCO
8. I2C0 only. Not supported on all GPIO Ports. Functionality limited to receive address recognition
9. Pin wake-up in EM4 supported only on GPIO_EM4WUx pins. Consult data sheet for complete list of pins.
10. If enabled in EMU->EM4CTRL.EM4IORETMODE.
11. Module is in Low Power Domain B (PD0B). The entire PD0B will be kept on in EM2/3 (resulting in higher current draw) if any module in PD0B is enabled on EM2/3 entry.

The different energy modes are summarized in the following sections.

12.3.1.1 EM0

EM0 provides all system features.

- Cortex-M33 is executing code
- Radio functionality is available
- High and low frequency clock trees are active
- All oscillators are available
- All peripheral functionality is available

12.3.1.2 EM1

EM1 disables the core but leaves the remaining system fully available.

- Cortex-M33 is in sleep mode. Clocks to the core are off
- Radio functionality is available
- High and low frequency clock trees are active
- All oscillators are available
- All peripheral functionality is available

12.3.1.3 EM1P

EM1P is a subset of EM1 which allows the radio to operate while the core and high-speed peripherals are shut down to save energy. It is entered when the radio is active and software requests to enter EM2.

- Cortex-M33 is in sleep mode. Clocks to the core and all high-speed peripherals are off
- Radio and HFXO remain active
- All peripherals and oscillators capable of EM2, EM3 or EM4 operation are available

12.3.1.4 EM2

This is the first level into the low power energy modes. Most of the high frequency peripherals are disabled or have reduced functionality. Memory and registers retain their values.

- Cortex-M33 is in sleep mode. Clocks to the core are off.
- Radio inactive
- RFSENSE available
- High frequency clock tree is inactive
- Low frequency clock tree is active
- The following oscillators are available
  - LFRCO, LFXO, ULFRCO, FSRCO (on demand, if used by the IADC)
- The following low frequency peripherals are available
  - RTCC, BURTC, WDOG, and LETIMER
- The following analog peripherals are available (with potential limitations on functionality)
  - IADC
- Wake-up to EM0 through
  - Peripheral interrupt, reset pin, power on reset, asynchronous pin interrupt, I2C0 address recognition, EUART0 interrupt, or RFSENSE
- Wake-up to EM1 through
  - Radio data transfer request
  - Part returns to EM2 when transfers are complete
- RAM and register values are preserved
  - RAM blocks may be optionally powered down for lower power
- GPIO pin state is retained
- RTCC memory is retained
- Debug connectivity is unavailable by default to reduce current consumption. Debug connectivity can be enabled by setting the EM2DBGEN bit in the EMU_CTRL register, and will consume about 0.5 uA extra supply current.

12.3.1.5 EM3

In this low energy mode, all low frequency oscillators (LFXO, LFRCO) and all low frequency clocks derived from them, are stopped, as well as all high frequency clocks. Most peripherals are disabled or have reduced functionality. Memory and registers retain their values.

- Cortex-M33 is in sleep mode. Clocks to the core are off.
- Radio inactive
- RFSENSE available
- High frequency clock tree is inactive
- All low frequency clock trees derived from the low frequency oscillators (LFXO, LFRCO) are inactive
- The following oscillators are available
  - ULFRCO, FSRCO (on demand, if used by the IADC)
- The following low frequency peripherals are available if clocked by the ULFRCO
  - RTCC, BURTC, and WDOG
- The following analog peripherals are available (with potential limitations on functionality)
  - IADC
- Wake-up to EM0 through
  - Peripheral interrupt, reset pin, power on reset, asynchronous pin interrupt, I2C0 address recognition, or RFSENSE
- Wake-up to EM1 through
  - Radio data transfer request
  - Part returns to EM3 when transfers are complete
- RAM and register values are preserved
  - RAM blocks may be optionally powered down for lower power
- GPIO pin state is retained
- RTCC memory is retained
- Debug connectivity is unavailable by default to reduce current consumption. Debug connectivity can be enabled by setting the EM2DBGEN bit in the EMU_CTRL register, and will consume about 0.5 uA extra supply current.

12.3.1.6 EM4

EM4 is the lowest energy mode of the part. There is no retention except for GPIO PAD state and BURAM values. Wake-up from EM4 requires a reset to the system, returning it back to EM0

- Cortex-M33 is off
- Radio is off
- RFSENSE available
- High frequency clock tree is off
- Low frequency clock tree may be active
- No RAM or register values are retained, except for the BURAM.
- The following oscillators are on if used by the BURTC:
  - LFRCO, LFXO, ULFRCO
- The following low frequency peripherals are available
  - BURTC
- Wake-up to EM0 through
  - BURTC interrupt, reset pin, power on reset, asynchronous pin interrupt (on GPIO_EM4WUx pins only), or RFSENSE
- GPIO pin state may be retained (depending on EMU->EM4CTRL.EM4IORETMODE configuration)

12.3.2 Entering Low Energy Modes

The following sections describe the requirements for entering the various energy modes.

12.3.2.1 Entry Into EM1

Energy mode EM1 is entered when the Cortex-M33 executes the Wait For Interrupt (WFI) or Wait For Event (WFE) instruction while the SLEEPDEEP bit the Cortex-M33 System Control Register is cleared. The MCU can re-enter sleep automatically out of an Interrupt Service Routine (ISR) if the SLEEPONEXIT bit in the Cortex-M33 System Control Register is set. Refer to ARM documentation on entering Sleep modes.

Alternatively, EM1 can be entered from either EM2 or EM3 due to certain peripheral wake-up requests, allowing transfers from the peripheral to system RAM. The system will return back to EM2 or EM3 once the peripheral has completed its transfers and processing.

12.3.2.2 Entry Into EM2 or EM3

Energy mode EM2 or EM3 may be entered when **all** of the following conditions are true:

- Radio state machine is in OFF state
- Cortex-M33 (if present) is in DEEPSLEEP state
- Flash Program/Erase Inactive
- DMA done with all current requests
- A debugger is not currently connected.

Energy mode EM2 is entered from EM0 when the Cortex-M33 executes the Wait For Interrupt (WFI) or Wait For Event (WFE) instruction while the SLEEPDEEP bit in the Cortex-M33 System Control Register is set. The MCU can re-enter DeepSleep automatically out of an Interrupt Service Routine (ISR) if the SLEEPONEXIT bit in the Cortex-M33 System Control Register is set. Refer to ARM documentation on entering Sleep modes.

Alternately, EM2 or EM3 is entered from EM1 upon the completion of a Peripheral Wake-Up Request from the RAC if no EM0 wake-up happens in the meantime.

When entering EM2 or EM3, if any peripheral on an auxiliary low power domain (PD0B, PD0C, etc.) is enabled, that auxiliary low power domain will be powered, causing higher current draw. Otherwise, the auxiliary power domain will be powered down. See 12.3.4 Power Domains for more information.

12.3.2.3 Entry Into EM4

Energy mode EM4 is entered through register access.

Software must ensure no modules are active, such as the Radio, when entering EM4.

Software may enter EM4 from EM0 by writing the sequence 2,3,2,3,2,3,2,3,2 to EM4CTRL->EM4ENTRY bit field. If the EM4BLOCK bit in WDOGn_CTRL is set, the CPU will be prevented from entering EM4 by software request.

An active debugger connection will prevent entry into EM4.

12.3.3 Exiting a Low Energy Mode

A system in EM2 and EM3 can be woken up to EM0 through regular interrupt requests from active peripherals. Since state and RAM retention is available, the EFR32 Series 2 is fully restored and can continue to operate as before it went into the Low Energy Mode.

Wake-up from EM4 is performed through a reset. Wake-up from a specific module must be enabled in that module's EM4WUEN register.

Enabled interrupts that can cause wake-up from EM2, EM3, and EM4 are shown in the following table. The wake-up triggers always return the device to EM0. Additionally, any reset source will return to EM0.

Table 12.3. Wake-Up Triggers from Low Energy Modes

Peripheral	Wake-Up Trigger	EM2	EM3	EM4
LETIMER	Any enabled interrupt	Yes	-	-
LFXO	Ready Interrupt	Yes	-	-
LFRCO	Ready Interrupt	Yes	-	-
WDOG	Any enabled interrupt	Yes	Yes	-
I^2C0	Receive address recognition	Yes	Yes	-
EUART0	Any enabled interrupt	Yes	-	-
RFSENSE	Wake condition met	Yes	Yes	Yes
RTCC	Any enabled interrupt	Yes	Yes	-
BURTC	Timeout	Yes	Yes	Yes [1]
EMU Temperature Sensor	Measured temperature outside the defined limits	Yes	Yes	-
Pin Interrupts	Transition	Yes[2]	Yes[2]	Yes [1 3]
Reset Pin	Assertion	Yes	Yes	Yes
Power	Cycle Off/On	Yes	Yes	Yes

Note:

1. Corresponding bit in the module's EM4WUEN must be set.
2. Available on Port A, Port B, and all EM4WU pins.
3. Only available on EM4WU pins.

12.3.4 Power Domains

The EFR32xG22 implements several independent power domains which are powered down to minimize supply current when not in use. Power domains are managed automatically by the EMU.

The lowest-energy power domain is the "high-voltage" power domain (PDHV), which supports extremely low-energy infrastructure and peripherals. Circuits powered from PDHV are always on and available in all energy modes.

The next power domain is the low power domain (PD0), which is further divided to power subsets of peripherals. All PD0 power domains are shut down in EM4. Circuits powered from PD0 power domains may be available in EM0, EM1, EM2, and EM3.

Low power domain A (PD0A) is the base power domain for EM2 and EM3 and will always remain on in EM0-EM3. It powers the most commonly-used EM2 and EM3-capable peripherals and infrastructure required to operate in EM2 and EM3. Auxiliary PD0 power domains (PD0B, PD0C, etc.) power additional EM2 and EM3-capable peripherals on demand. If any peripherals on one of the auxiliary power domains is enabled, that power domain will be active in EM2 and EM3. Otherwise, the auxiliary PD0 power domains will be shut down to reduce current.

Note: The number of PD0 power domains varies depending on the device family. All devices support at least the base PD0 power domain (PD0A). Refer to the device data sheet for information on the assigment of peripherals to auxiliary PD0 power domains.

The active power domain (PD1) powers the rest of the device circuitry, including the CPU core and EM0 / EM1 peripherals. PD1 is always powered on in EM0 and EM1. PD1 is always shut down in EM2, EM3, and EM4.

12.3.5 Voltage Scaling

The EFR32xG22 supports supply voltage scaling for the LDO powering DECOUPLE. Voltage scaling helps to optimize the energy efficiency of the system by operating at lower voltages when possible. Three supply voltage operating points are available:

Table 12.4. Voltage Scaling Options

VSCALE Setting	DECOUPLE Voltage	Operating Conditions
VSCALE2	1.1 V	EM0/EM1 Operation up to 80 MHz EM2 and EM3
VSCALE1	1.0 V	EM0/EM1 Operation up to 40 MHz EM2 and EM3
VSCALE0	0.9 V	EM2 and EM3 Only

12.3.5.1 Voltage Scaling in EM0 and EM1

In EM0 and EM1, the voltage scaling value should be set according to the desired operating frequency. The system defaults to VSCALE2 out of reset. To operate above 40 MHz, VSCALE2 should always be used. If the system will operate below 40 MHz, VSCALE1 may be used to save energy.

The voltage scaling value for EM0 and EM1 is changed via software command bits in the EMU_CMD register. Setting EMU_CMD_EM01VSCALE1 will switch to VSCALE1, and setting EMU_CMD_EM01VSCALE2 will switch to VSCALE2.

The command initiates a voltage change operation, but some time is needed before the new supply voltage is reached. When changing between VSCALE values in EM0, it takes approximately 150 us to ramp the voltage down and approximately 32 us to ramp the voltage up to the new values (see the data sheet specifications for exact numbers). During this time, SRAM access is prohibited by the hardware and any accesses to SRAM from the CPU or DMA will be blocked until the operation is complete. The EMU_STATUS_VSCALEBUSY bit indicates when a voltage scale change is in progress. When the operation is complete the EMU_IF_VSCALEDONEIF flag will be set.

Note: Because SRAM access is blocked during a voltage scaling operation, it is recommended to configure the desired EM0 / EM1 voltage scaling once during initial boot-up for systems operating at VSCALE1.

The current VSCALE setting can be read at any time from the EMU_STATUS_VSCALE field.

12.3.5.2 Voltage Scaling in EM2 and EM3

A separate voltage scaling value is used during EM2 and EM3. This allows the core to run at a higher voltage when in EM0 / EM1 and reduce the voltage in EM2 and EM3 for power savings. The voltage scale level for EM2 and EM3 is set using the EMU_CTRL_EMU23VSCALE field. The new voltage scaling level will be applied when the system is in EM2 or EM3, and return to the EM0 / EM1 voltage scaling level automatically when the system exits the low energy mode. EMU_CTRL_EMU23VSCALE should always be set to a VSCALE level at or lower than the one used in EM0/EM1. For example, if EM0 / EM1 voltage scaling is set to VSCALE1, then EM2 / EM3 voltage scaling can be set to VSCALE1 or VSCALE0.

If the voltage scaling level for EM0 / EM1 is lower than the level set for EM2 / EM3, additional time is needed to wake up from the low powered state (see the device data sheet for specific timing). The lowest sleep current will be obtained by setting EMU23VSCALE to VSCALE0, and the fastest wake times will be obtained when EMU23VSCALE is set to the EM0 / EM1 voltage scaling value.

12.3.6 EM0 / EM1 Peripheral Register Retention

When the device enters EM2 or EM3, all peripherals will retain their register configurations by default. Retention for peripherals on the PD1 power domain (i.e. those which do not operate in EM2 and EM3), can optionally be disabled by setting bit 0 of the EMU_PD1PARETCTRL_PD1PARETDIS field. Disabling retention reduces the supply current in EM2 and EM3 slightly. However, the peripheral register interfaces will be reset upon exit to EM0.

Important: This feature is not currently supported by Silicon Labs software stacks. It is the responsibility of the user software to reconfigure any peripherals as necessary when the device wakes to EM0.

12.3.7 Power Configurations

In order to provide the lowest power solutions, the EFR32xG22 comes with a DC-DC module to power internal circuits. The EFR32xG22 may be operated with or without the DC-DC. When used, the DC-DC requires an external inductor and capacitor (refer to the data sheet for recommended values).

The EFR32xG22 has multiple power supply rails: a DC-DC regulator input (VREGVDD), IO Supply (IOVDD), Analog (AVDD), RF Analog Supply (RFVDD), RF Power Amplifier Supply (PAVDD), Digital LDO and flash (DVDD), and Low Voltage Digital Supply (DECOUPLE). Additional detail for each configuration and option is given in the following sections.

Due to on-chip circuitry (e.g., diodes), some EFR32 power supply pins have a dependent relationship with one or more other power supply pins. These internal relationships between the external voltages applied to the various EFR32 supply pins are defined below. Exceeding the below constraints can result in damage to the device and/or increased current draw.

- VREGVDD >= DVDD

  Note: In systems not using the DC-DC converter, VREGVDD must be shorted to DVDD external to the device.

- PAVDD >= RFVDD
- DVDD >= DECOUPLE
- AVDD, IOVDD: No dependency with other supply pins.

Additionally, there are other system-level considerations when assigning power supplies.

- The usable range for analog signals connected to GPIO (such as IADC inputs) will be limited to the lower of AVDD and IOVDD.
- The RESETn pin has an internal pullup to the DVDD supply. If RESETn is driven by external circuitry above DVDD, additional current may flow into the pin due to this pullup.

12.3.7.1 Power Configuration 0: STARTUP

Upon power-on reset (POR), the system is configured in a safe Startup Configuration that supports all of the available Power Configurations. The Startup Configuration is shown in the simplified diagram below.

In the Startup configuration the DC-DC converter's Bypass switch is ON (i.e., the VREGVDD pin is shorted internally to the DVDD pin).

After power on, firmware can elect to turn on the DC-DC if the external hardware configuration supports it.

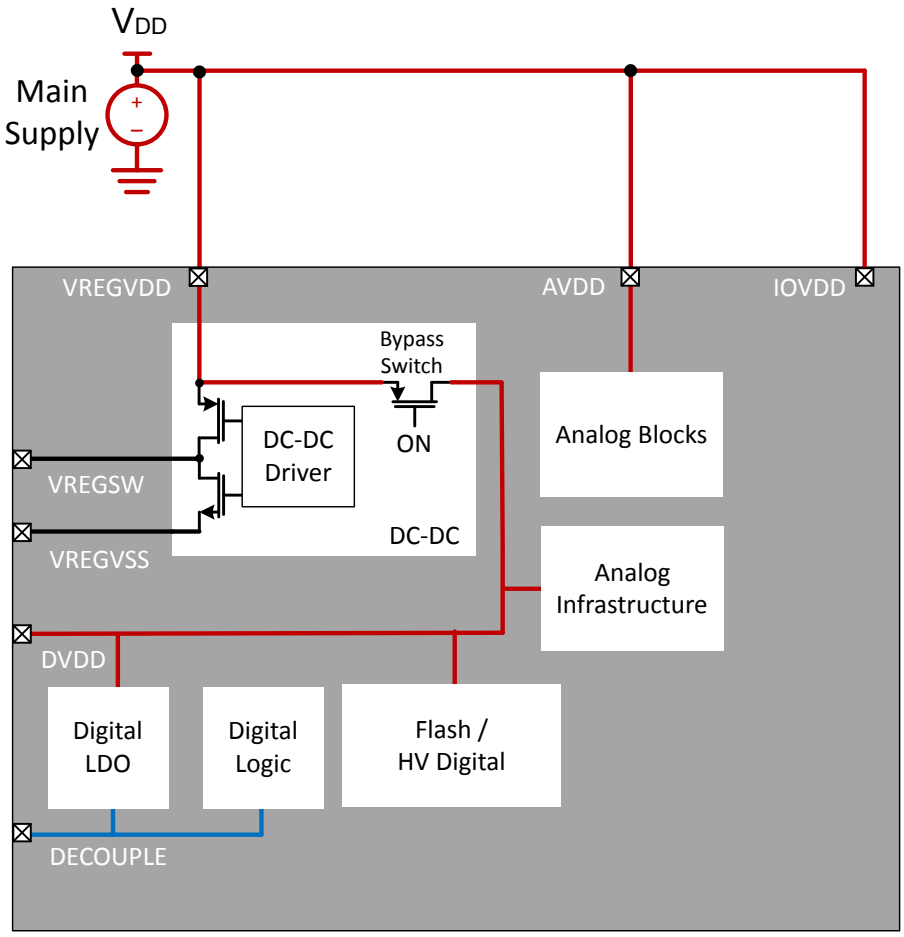


Figure 12.2. Startup Power Configuration

12.3.7.2 Power Configuration 1: No DC-DC

In Power Configuration 1, the DC-DC converter is unused, and all power is supplied by external sources. The DVDD pin must be shorted to VREGVDD.

IOVDD, AVDD, RFVDD and PAVDD may be supplied by the same supply as VREGIN and DVDD (as shown in 12.3.7.2 Power Configuration 1: No DC-DC), or they may be powered from a separate source.

VREGSW must be left disconnected in this configuration.

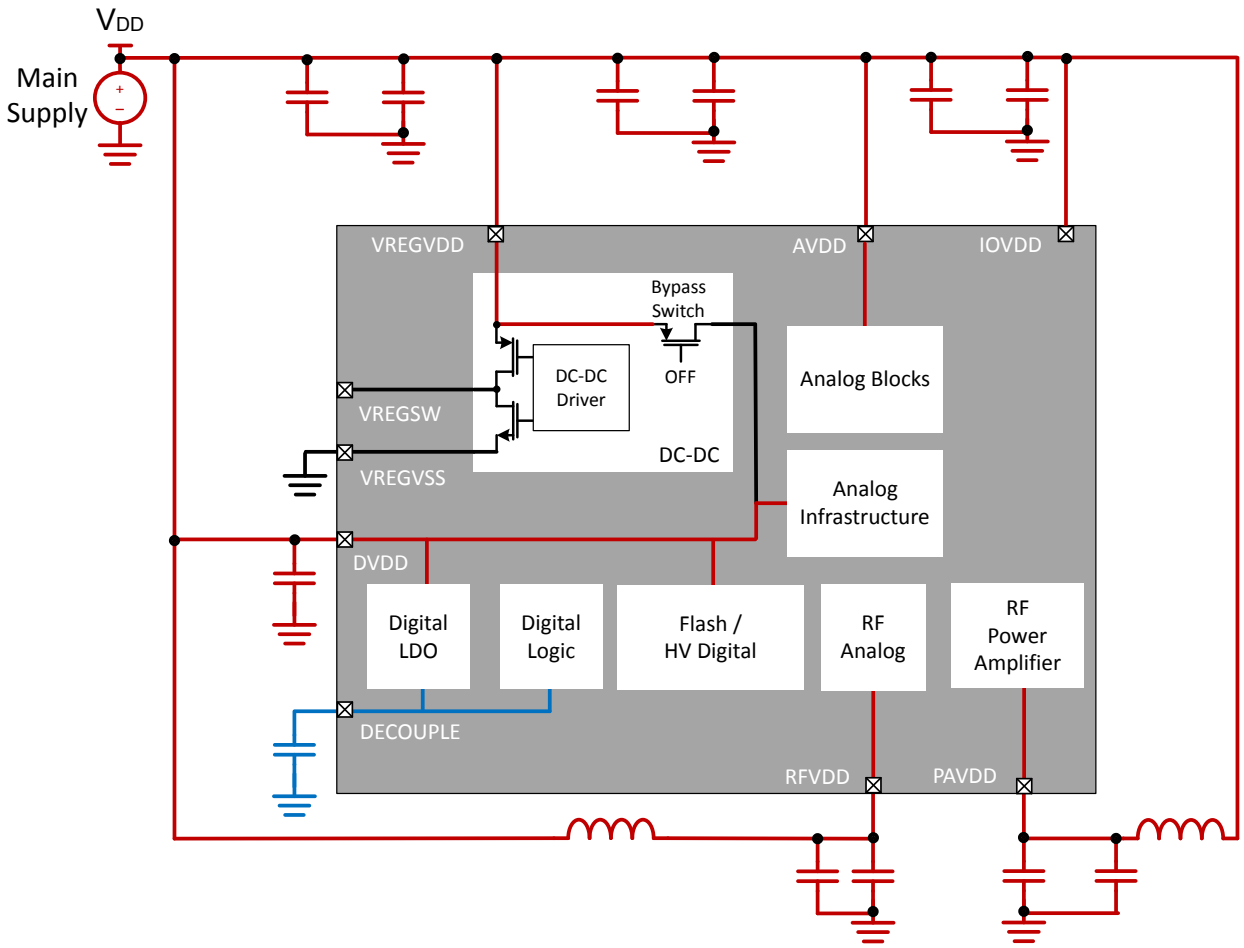


Figure 12.3. DC-DC Off Power Configuration

12.3.7.3 Power Configuration 2: DC-DC

For the lowest power applications, the DC-DC converter can be used to power the rest of the supplies on the device. When the DC-DC converter is used to regulate the voltage at DVDD, the maximum supply voltage may be limited by the operating temperature and/or the average lifetime load conditions. Refer to the device datasheet for additional details.

In Power Configuration 2, the DC-DC Output (V_{DCDC}) is connected to DVDD and optionally, to all the other supplies on the chip. In the configuration shown in Figure 12.4 DC-DC Power Configuration on page 298, the AVDD and IOVDD supplies are connected to the main supply to support higher voltage external interfaces.

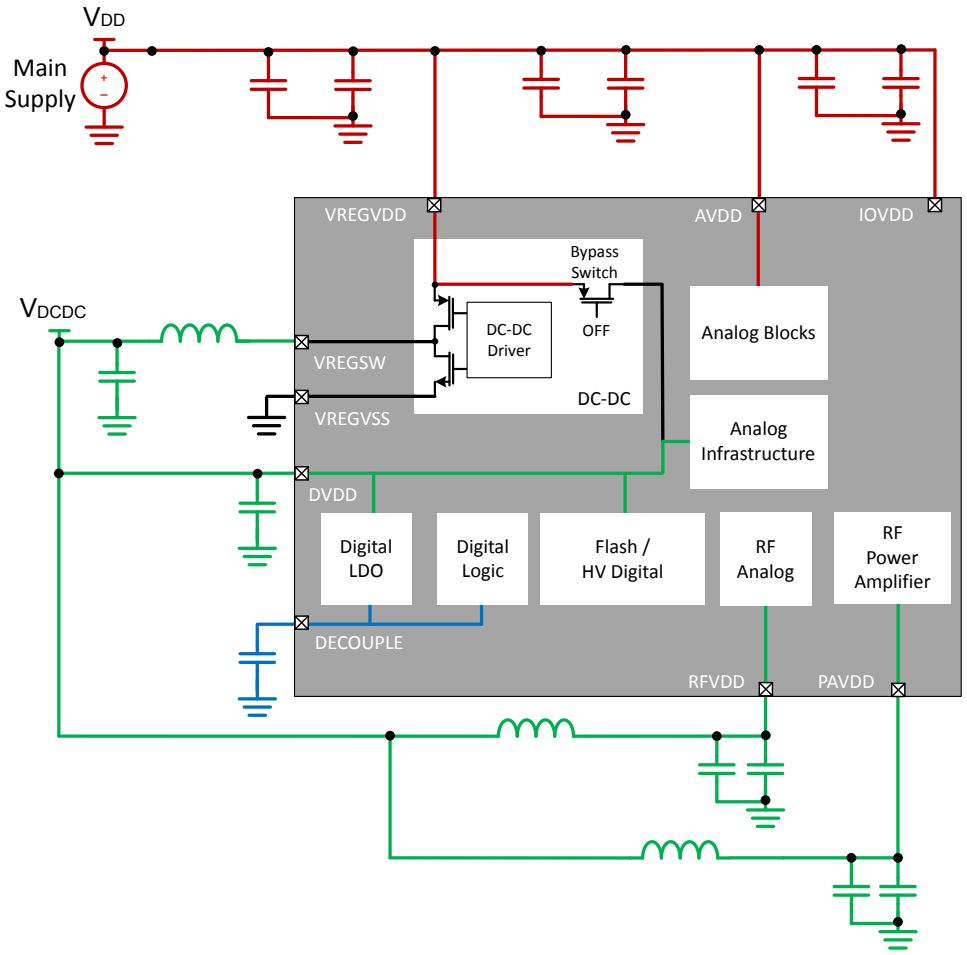


Figure 12.4. DC-DC Power Configuration

As the Main Supply voltage approaches the DC-DC output voltage, it eventually reaches a point where becomes inefficient (or impossible) for the DC-DC module to regulate V_{DCDC}. At this point, firmware can enable bypass mode, which effectively disables the DC-DC and shorts the Main Supply voltage directly to the DC-DC output. If and when sufficient voltage margin on the Main Supply returns, the system can be switched back into DC-DC regulation mode.

12.3.8 DC-to-DC Interface

The EFR32xG22 devices feature a DC-DC buck converter which requires a single external inductor and a single external capacitor. The input supply is the VREGVDD pin, and the DC-DC converter will produce a nominal 1.8 V output at the DVDD pin to power radio and MCU functions. The DC-DC converter is an efficient PFM (Pulse Frequency Modulation) architecture, delivering up to 60 mA of current. In addition, the DC-DC converter supports an unregulated bypass mode, in which the input voltage is directly shorted to the DC-DC output. An integrated programmable supply monitor and dedicated interrupt allows software to enable the bypass switch when the VREGVDD supply voltage is below the minimum allowable voltage for the output current load.

The input supply VREGVDD has a maximum range between 1.8 V and 3.8 V, but is limited by application parameters, including transient current load, operating junction temperature, and the lifetime average current load.

Refer to the device datasheet for more details on the input supply voltage range.

12.3.8.1 Bypass Mode and VREGVDD Comparator

In bypass mode, the VREGVDD input voltage is directly shorted to the DC-DC converter output through an internal switch. Bypass mode is enabled automatically during a power-on-reset. Bypass mode can also be enabled and disabled through software, using the DCDC_CTRL_MODE field. When set to BYPASS, the bypass switch is enabled and DC-DC regulation will be disabled. Consult the data sheet for the bypass switch impedance specification.

The EFR32xG22 includes a supply comparator circuit to help software determine when the VREGVDD supply is high enough to enable DC-DC regulation or when to change to bypass mode. The THRESSEL field in the EMU_VREGVDDCMPCTRL register sets the comparator threshold between 2.0 and 2.3 V, and the VREGINCMPEN bit is used to enable the supply comparator. When the VREGVDD comparator is used, DCDC_STATUS_VREGIN can be read by software to determine whether VREGVDD is above or below the established threshold.

The VREGVDD comparator can also generate interrupt events when the input supply is above or below the specified threshold. The VREGINHIGHIEN and VREGINLOWIEN bits in DCDC_IEN are used to enable the above / below threshold interrupts, respectively. The VREGVDD comparator will be active and generate interrupts in EM0 and EM1 only.

The VREGVDD Comparator status is always captured and stored in RMURSTCAUSE.VREGIN on any reset event, even if the reset is not caused by VREGVDD being too low. At startup, the firmware should determine if the last reset was caused by a low VREGVDD condition by checking the following:

EMU_RSTCAUSE_VREGIN & (EMU_RMURSTCAUSE_DVDDBOD | EMU_RMURSTCAUSE_DVDDLEBOD)

If true, the part should remain in bypass mode with the DCDC disabled.

12.3.8.2 DC-DC Startup

Out of power-on-reset (POR), the DC-DC converter defaults to bypass mode and the DC-DC block is disabled. Before enabling the DC-DC, software should first configure and enable the VREGVDD comparator. Once the thresholds for the VREGVDD comparator have been configured and the comparator enabled, the DCDC_STATUS_VREGIN bit should be checked to ensure that the input supply is above the threshold. When the input supply is sufficient, the DC-DC may be configured and enabled. The following steps outline this procedure:

1. Set VREGVDD comparator threshold with EMU_VREGVDDCMPCTRL_THRESSEL
2. Enable VREGVDD comparator with EMU_VREGVDDCMPCTRL_VREGINCMPEN
3. Check DCDC_STATUS_VREGIN:
   - If low, VREGIN is above the programmed threshold and it is safe to enter DC-DC mode
   - If high, VREGIN is below the programmed threshold and firmware should remain in bypass mode
4. Enable the DC-DC module with DCDC_EN_EN = 1
5. Configure the IPKVAL and DRVSPEED settings in DCDC_EM01CTRL0 and DCDC_EM23CTRL0.
6. Enable any required interrupts via DCDC_IEN.
7. Start the DC-DC by setting DCDC_CTRL_MODE to DCDCREGULATION.

The DC-DC will enter a warmup phase for approximately 100 us, then disable the bypass switch and begin using the DC-DC core to regulate the output voltage. The DCDC_IF_RUNNINGIF interrupt flag will indicate when the switch from bypass to DC-DC is complete, however this does not indicate that the output is regulated. Until the output capacitor discharges due to normal current draw from the system, the voltage may be higher than 1.8 V. The DCDC_IF_REGULATIONIF interrupt flag will indicate when the DC-DC has reached regulation and is providing the desired output voltage.

If the VREGINLOWIF interrupt occurs, software should immediately switch back to bypass mode by clearing DCDC_CTRL_MODE to BYPASS.

12.3.8.3 Recommended Configuration Settings

Certain DC-DC parameters are adjustable for fine-tuning of performance, but the majority of applications will not need to use any other than the recommended settings. All datasheet parameters are specified using the recommended settings detailed in this section. The configuration settings must be set before DC-DC regulation is started, and must not be changed while the DC-DC is active.

The DCDC_EM01CTRL0 and DCDC_EM23CTRL0 registers each have an IPKVAL field to adjust the maximum peak / load current, and a DRVSPEED field to adjust the driver speed. DCDC_EM01CTRL0 sets the configuration for EM0 and EM1 operation while DCDC_EM23CTRL0 sets the configuration for EM2 and EM3 operation. The DCDC_CTRL_IPKTMAXCTRL field adjusts the maximum time for peak current detection, which impacts the voltage ripple at the DC-DC output. The recommended settings are shown in Table 12.5 DRVSPEED, IPKVAL, and IPKMAXCTRL Recommended Settings on page 300.

Table 12.5. DRVSPEED, IPKVAL, and IPKMAXCTRL Recommended Settings

Bit Field	Recommended Setting
EM01CTRL0_IPKVAL	9 (LOAD60MA)
EM01CTRL0_DRVSPEED	1 (DEFAULT_SETTING)
EM23CTRL0_IPKVAL	3 (LOAD5MA)
EM23CTRL0_DRVSPEED	3 (BEST_EFFICIENCY)
DCDC_CTRL_IPKTMAXCTRL	4 (TMAX_1P19US)

The DCDC_CTRL_DCMONLYEN field can be used to select between DCM and CCM mode. The default setting (1) is to use DCM mode, and should not be changed for most applications.

12.3.8.4 EM4 Entry

The DC-DC is available in all energy modes except for EM4. If the system wants to enter EM4, the DC-DC converter must first be turned off and switched over to bypass mode. The system will not enter EM4 if the DC-DC is active. If an attempt is made to go into EM4 with DC-DC active, it will be blocked, and the DCDC_IF_EM4ERR flag will be set.

12.3.9 EFP Communication

EFR32xG22 has built-in hardware support for the EFP01 Energy Friendly PMICs. The EFP01's dual-DCDC converter outputs can, for example, provide power to both the 1.8V (e.g., DVDD/AVDD/IOVDD) supplies as well as the 1.1V (DECOUPLE) for improved efficiency. Consult EFP01 Datasheet for detailed information.

EFR32xG22 communicates with EFP01 via I2C. EFP01 also has a unidirectional, open-drain IRQ# output to EFR32xG22 to indicate status flag changes.

EFR32xG22's EFP01 hardware support must be enabled by setting one (or both) of the EFPDRVDECOUPLE or the EFPDRVDVDD bits in the EMU_CTRL register:

1. EFPDRVDECOUPLE: Set this bit if EFP01's DCDC output will be powering EFR32xG22's DECOUPLE supply. Once set, EFR32xG22's internal LDOs will be disabled, and the EMU will control EFP01 for voltage-scaling (if enabled). In addition, (if connected as described below) any interrupts received by EFP01's IRQ will set the EFPIF flag in the EMU_EFPIF register. Note that because this bit disables in the internal LDO's powering the core, it should be set until after EFP01's DECOUPLE output has been configured and enabled.
2. EFPDRVDVDD: Set this bit if EFP01's DCDC output is powering EFR32xG22's DVDD supply (or DVDD along with other 1.8V supply inputs). This mode assumes that EFR32xG22's internal DCDC is not being used, so the VREGVDD and PAVDD pins should be shorted together on the PCB. In addition, (if connected as described above) any interrupts received by EFP01's IRQ will set the EFPIF flag in the EMU_EFPIF register.

EFR32xG22 provides a dedicated hardware IRQ vector for the EFP01's IRQ output. To use the EFR32xG22's hardware support for EFP01's IRQ output:

1. The PC5 pin should be configured as an input and connected on the PCB to EFP01's IRQ pin. Note that although this pin exists on Port C, which typically doesn't support EM2/3 operation, when used as a EFP01 IRQ input the PC5 pin can operate in EM2/3. For EM4 operation, PC5 would need to be configured as a EM4Wakeup.
2. EFP Hardware support must be enabled by setting either the EFPDRVDECOUPLE or the EFPDRVDVDD bits as described above.

Once enabled, the EFP interupt flag in the EMU_EFPIF register will be set whenever the EFP01 IRQ line goes low. A processor interupt can be generated by setting the EFPIEN bit in the EMU_EFPIEN register.

EFR32xG22 supports EFP01's optional Direct Mode interface to allow fast-energy mode transitions into and out of all energy modes (EM0/1, EM2/3, EM4). Ordinarilly, I2C transactions are used to toggle EFP01's energy mode state - however, a single I2C tranaction can take over 100usec. In Direct Mode, the EFR32xG22 retasks the I2C pins as push-pull outputs with pull-ups disabled to control the EFP01's energy mode state directly, allowing much faster energy mode transistions. State definitions are defined in Table 12.6 Direct Mode Energy Mode States on page 301. Because the Direct Mode feature is non-I2C compliant, it should be enabled only during periods when no communciation between EFR32xG22 and EFP01 is required. It is recommended for firmware to wait for the I2C STOP interrupt ensure no I2C transaction is in progress before switching to Direct Mode.

Table 12.6. Direct Mode Energy Mode States

Direct Mode State	I2C SCL Level	I2C SDA Level	Allowed State Transitions
EM0	1	1	• EM2 • I^2C Start Condition
EM2	0	1	• EM0 • EM4
EM4	0	0	• EM2[2]
I^2C Start Condition	1	0	

1 Direct mode transitions between EM0 and EM4 are not allowed. The system must briefly go through the EM2 state on EM4 exit or entrance.

2 Direct mode transitions between EM0 and EM4 are not allowed. The system must briefly go through the EM2 state on EM4 exit or entrance.

To enable Direct Mode:

1. The I2C1 module must be used to communicate with EFP01. I2C0 is not supported.
2. The I2C1_SDA function must be routed to the PC1 pin and connected on the PCB to EFP01's I2C_SDA pin.
3. The I2C1_SCL function must be routed to the PC2 pin and connected on the PCB to EFP01's I2C_SCL pin.

4. Direct Mode must be enabled by setting the EFPDIRECTMODEN bit in the EMU_CTRL register. The EMU will automatically disable the I2C1 internal pull-ups when in Direct Mode
5. EFP Hardware support must be enabled by setting either the EFPDRVDECOUPLE or the EFPDRVDVDD bits as described above.

12.3.10 Brown Out Detector (BOD)

Brown out detectors ensure that the minimum supply required for the chip to operate properly and safely is provided to the EFR32xG22. Once triggered, a BOD will generate a system reset.

Each BOD raw output is visible via the EMU_ANASTATUS register and can also be routed to a GPIO via PRS for observability (see EMU PRS section for more details). In addition the xxxxMASKONCFGCHG status bits (also in the EMU_ANASTATUS register) indicate when a particular BOD is being masked by HW following a configuration change (enable/disable or trim change).

All BODs detect when the supply falls below a programmed threshold except DECOVMBOD (Over Voltage Monitoring),which detects when the supply goes above a predefined threshold.

All BOD can be individually enabled and masked. EMU_TESTCTRL_BODMASK can be used to mask all BOD's at once.

Table 12.7. EFR32xG22 BODs

BOD	Control Register	Supported Energy Modes	Function
DVDDBOD	EMU_DVDDBOD	EM0/1	Monitors the DVDD supply in EM0 and EM1. Hardware enables this BOD automatically in EM0/EM1 and disables it in EM2/EM3/EM4
DVDDLEBOD	EMU_DVDDLEBOD	EM2/3/4	Low Energy BOD monitors the DVDD supply in EM2/EM3/ EM4. DVDDLEBOD is automatically masked by hardware for ~100us after it is enabled to allow it to settle
DECBOD	EMU_DECBOD	EM0/1/2/3	Monitors the DECOUPLE supply. DECBOD is automatically masked by hardware for ~20us after it is enabled to allow it to settle.
DECOVMBOD	EMU_DECBOD	EM0/1/2/3	Monitors the DECOUPLE supply Over Voltage by detecting DECOUPLE going over a specified threshold. DECOVMBOD is automatically masked by hardware for ~20us after it is enabled to allow it to settle.
AVDDBOD	EMU_BOD3SENSE	EM0/1/2/3/4	Monitors the AVDD supply. Automatically masked by hardware for ~100us after it is enabled to allow it to settle.
IOVDDBOD	EMU_BOD3SENSE	EM0/1/2/3/4	Monitors the IOVDD supply. Automatically masked by hardware for ~100us after it is enabled to allow it to settle. (Note that some devices may have multiple IOVDD supplies.)

12.3.11 Reset Management Unit

EMU RMU (Reset Management Unit) ensures correct reset operation. It is responsible for connecting the different reset sources to the reset lines of the EFR32xG22. After reset, the M33 loads the stack pointer and program entry point from memory and start execution.

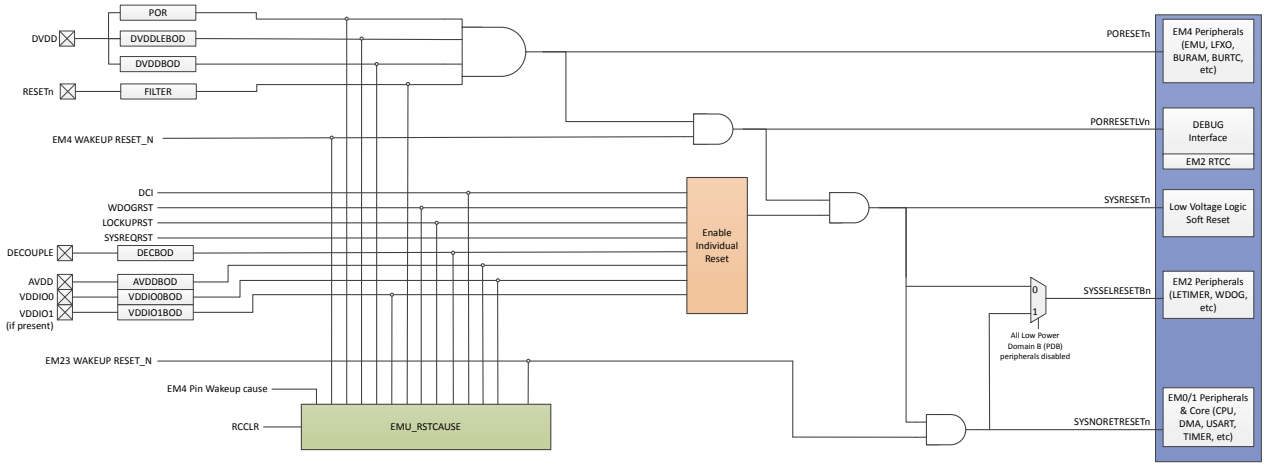


Figure 12.5. Reset Tree

There are two types of reset:

- HARD resets. Resets the entire chip. After a hard reset, the EFR32xG22 goes through its power up sequence. The time to return to EM0 mode can be a few hundred usec.
- SOFT resets. Resets only some of the digital low voltage logic. Resets the MCU subsystems and peripherals but doesn't affect digital HV logic (e.g., Power control, BURTC). The time to return to EM0 mode is on the order of a few usec.

EFR32xG22 Reset sources

- Power-on Reset (POR)
  - The POR ensures that EFR32xG22 does not start up before the supply voltage DVDD has reached the threshold voltage VPORthr (see Device Datasheet Electrical Characteristics for details). Before the threshold voltage is reached, EFR32xG22 is kept in reset state.
- RESET pin Reset
  - The RESETn pin includes an on-chip pull-up resistor to the DVDD supply, and can therefore be left unconnected if no external reset source is needed. Also connected to the RESETn line is a filter which prevents glitches from resetting the EFR32xG22.
- EM4 wakeup
  - System reset following EM4 exit.
- Watchdog reset
  - The Watchdog circuit is a timer which (when enabled) must be cleared by software regularly. If software does not clear it, a Watchdog reset is activated. This functionality provides recovery from a software stalemate. Refer to the Watchdog section for specifications and description.
- Core lockup condition
  - A MCU lockup is the result of the core being locked up because of an unrecoverable exception following the activation of the processor's built-in system state protection hardware.
- Software triggered reset
  - Software may initiate a reset (e.g. if it finds itself in a non-recoverable state). By asserting the SYSRESETREQ in the Application Interrupt and Reset Control Register, a reset is issued.
- Brown-Out Detection (BOD)
  - EFR32xG22 has multiple built in Brown-out detection (BOD) circuits, which monitor supply voltage level during operation. BOD circuits compare supply voltage to a programmed threshold level and issue a reset request when triggered.
- Debug Challenge Interface (DCI)

Whether a reset source trigger event lead to a system reset can be controlled via EMU_RMUCTRL register.
EMU_RMURSTCAUSE register

User can determine the cause of the last reset by querying the EMU_RMURSTCAUSE register. Once read, EMU_RMURSTCAUSE should be cleared via EMU_CMD_RCCLR.

Table 12.8. Reset Sources Summary

RSTCAUSE Bit	Name	Type	Can be Disabled?	Description
0	POR	Hard	No	Power On Reset.
1	PIN	Hard	No	Pin Reset.
2	EM4	Soft	No	EM4 Wakeup
3	WDOG0	Soft	Yes	Watchdog 0
5	LOCKUP	Soft	Yes	M33 Lockup
6	SYSREQ	Soft	Yes	M33 Core System Reset
7	DVDDBOD	Hard	No	DVDD BOD
8	DVDDLEBOD	Hard	No	DVDD LEBOD
9	DECBOD	Hard	Yes	DECOUPLE BOD
10	AVDDBOD	Soft	Yes	AVDD BOD
11	VDDIO0BOD	Soft	Yes	IOVDD 0 BOD
16	DCI	Soft	Yes	Debug Challenge Interface

12.3.12 Temperature Sensor

EMU provides a low energy periodic temperature measurement. A temperature measurement is taken once every 250 ms, with the 9-bit result stored in TEMP bit-field in EMU_TEMP register. The temperature value is expressed in degrees Kelvin. EMU_TEMP_TEMPLSB represents the measured temperature fractional part (in ¼ degree Kelvin).

Note: The EMU temperature sensor is always periodically taking single temperature measurements, except in EM4 (shutoff) mode.

To obtain better noise resolution, the temperature sensor also implements a hardware averaging function, and averaged results can be requested using the EMU_CMD_TEMPAVGREQ command. When TEMPAVGREQ is set by software, the temperature sensor will take 16 or 64 samples as quickly as possible. The TEMPAVGNUM field in EMU_CTRL determines how many temperature measurements will be averaged. The averaged result is stored in the 11-bit field EMU_TEMP_TEMPAVG, which represents the full temperature with resolution of ¼ degree Kelvin.

The EMU provides the following features around temperature changes:

- Interrupt when temperature is updated (EMU_IF_TEMP)
- Interrupt when averaged temperature result is updated (EMU_IF_TEMPAVG)
- Interrupt from LOW level trip (generate interrupt EMU_IF_TEMPLOWIF when measured temperature in EMU_TEMP_TEMP is below programmed threshold EMU_TEMPLIMITS_TEMPLOW)
- Interrupt from HIGH level trip (generate interrupt EMU_IF_TEMPHIGHIF when measured temperature in EMU_TEMP_TEMP is above programmed threshold EMU_TEMPLIMITS_TEMPHI)

High and Low thresholds are specified as 9-bit degree Kelvin values and compared against the single temperature result (EMU_IF_TEMP).

Measured temperature can be converted to degrees Celsius by subtracting 273.15 ($T_{Celsius} = T_{Kelvin} - 273.15$).

12.3.12.1 Linearization, Offset Correction, and Calibration

The raw value reported by the EMU temperature sensor follows a predictable curve. The output may be linearized and the systematic offset removed to achieve die temperature readings with better than +/- 2.5 degrees C accuracy over the full operating temperature range. Further accuracy can be achieved using in-system calibration.

To linearize the measurement and correct for the systematic offset, a second or third-order polynomial equation representing the nominal curve is used. Polynomial coefficients for both cases are shown in Table 12.9 Polynomial Coefficients on page 305. Note that the polynomial coefficients provided assume the raw output (in Kelvin) has been converted to Celsius prior to linearization.

Table 12.9. Polynomial Coefficients

Polynomial Order	x^3 Term	x^2 Term	x^1 Term	x^0 Term
Third Order	-8.186E-7	-3.005E-5	1.015	-2.860
Second Order	n/a	-1.570E-4	1.017	-2.733

Additional accuracy may be achieved by performing in-system calibration at known temperatures and operating conditions.

12.3.13 Register Resets

Each EMU register requires retaining state in various energy modes and power transitions and will consequently need to be reset with a different condition. The following reset conditions will apply to the appropriate set of registers as marked in the Register Description table.

- Reset with POR or Hard Pin Reset
- Reset with POR, Hard Pin Reset, or any BOD reset
- Reset with SYSEXTENDEDRESETn
- Reset with FULLRESETn (default)

If a register field is not marked with a specific reset condition then it is assumed to be reset with FULLRESETn.

12.3.14 Register Locks

EMU EMU_LOCK (for user accessible registers) can be used to control access to the EMU_RMUCTRL, EMU_CTRL, and EMU_DECBOD registers. The DCDC_LOCK register can be used to control access to the DC-DC registers.

12.4 EMU Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x010	EMU_DECBOD	RW	DECOUPLE LVBOD Control register
0x020	EMU_BOD3SENSE	RW	BOD3SENSE Control register
0x03C	EMU_VREGVDDCMPCTRL	RW	DC-DC VREGVDD Comparator Control Register
0x040	EMU_PD1PARETCTRL	RW	PD1 Partial Retention Control
0x060	EMU_LOCK	W	EMU Configuration lock register
0x064	EMU_IF	RWH INTFLAG	Interrupt Flags
0x068	EMU_IEN	RW	Interrupt Enables
0x06C	EMU_EM4CTRL	RW	EM4 Control
0x070	EMU_CMD	W	EMU Command register
0x074	EMU_CTRL	RW	EMU Control register
0x078	EMU_TEMPLIMITS	RW SYNC	EMU Temperature thresholds
0x084	EMU_STATUS	RH	EMU Status register
0x088	EMU_TEMP	RH SYNC	Temperature
0x090	EMU_RSTCTRL	RW	Reset Management Control register
0x094	EMU_RSTCAUSE	RH	Reset cause
0x0A0	EMU_DGIF	RWH INTFLAG	Interrupt Flags Debug
0x0A4	EMU_DGIEN	RW	Interrupt Enables Debug
0x100	EMU_EFPIF	RWH INTFLAG	EFP Interrupt Register
0x104	EMU_EFPIEN	RW	EFP Interrupt Enable Register
0x1010	EMU_DECBOD_SET	RW	DECOUPLE LVBOD Control register
0x1020	EMU_BOD3SENSE_SET	RW	BOD3SENSE Control register
0x103C	EMU_VREGVDDCMPCTRL_SET	RW	DC-DC VREGVDD Comparator Control Register
0x1040	EMU_PD1PARETCTRL_SET	RW	PD1 Partial Retention Control
0x1060	EMU_LOCK_SET	W	EMU Configuration lock register
0x1064	EMU_IF_SET	RWH INTFLAG	Interrupt Flags
0x1068	EMU_IEN_SET	RW	Interrupt Enables
0x106C	EMU_EM4CTRL_SET	RW	EM4 Control
0x1070	EMU_CMD_SET	W	EMU Command register
0x1074	EMU_CTRL_SET	RW	EMU Control register
0x1078	EMU_TEMPLIMITS_SET	RW SYNC	EMU Temperature thresholds
0x1084	EMU_STATUS_SET	RH	EMU Status register
0x1088	EMU_TEMP_SET	RH SYNC	Temperature
0x1090	EMU_RSTCTRL_SET	RW	Reset Management Control register
0x1094	EMU_RSTCAUSE_SET	RH	Reset cause

Offset	Name	Type	Description
0x10A0	EMU_DGIF_SET	RWH INTFLAG	Interrupt Flags Debug
0x10A4	EMU_DGIEN_SET	RW	Interrupt Enables Debug
0x1100	EMU_EFPIF_SET	RWH INTFLAG	EFP Interrupt Register
0x1104	EMU_EFPIEN_SET	RW	EFP Interrupt Enable Register
0x2010	EMU_DECBOD_CLR	RW	DECOUPLE LVBOD Control register
0x2020	EMU_BOD3SENSE_CLR	RW	BOD3SENSE Control register
0x203C	EMU_VREGVDDCMPCTRL_CLR	RW	DC-DC VREGVDD Comparator Control Register
0x2040	EMU_PD1PARETCTRL_CLR	RW	PD1 Partial Retention Control
0x2060	EMU_LOCK_CLR	W	EMU Configuration lock register
0x2064	EMU_IF_CLR	RWH INTFLAG	Interrupt Flags
0x2068	EMU_IEN_CLR	RW	Interrupt Enables
0x206C	EMU_EM4CTRL_CLR	RW	EM4 Control
0x2070	EMU_CMD_CLR	W	EMU Command register
0x2074	EMU_CTRL_CLR	RW	EMU Control register
0x2078	EMU_TEMPLIMITS_CLR	RW SYNC	EMU Temperature thresholds
0x2084	EMU_STATUS_CLR	RH	EMU Status register
0x2088	EMU_TEMP_CLR	RH SYNC	Temperature
0x2090	EMU_RSTCTRL_CLR	RW	Reset Management Control register
0x2094	EMU_RSTCAUSE_CLR	RH	Reset cause
0x20A0	EMU_DGIF_CLR	RWH INTFLAG	Interrupt Flags Debug
0x20A4	EMU_DGIEN_CLR	RW	Interrupt Enables Debug
0x2100	EMU_EFPIF_CLR	RWH INTFLAG	EFP Interrupt Register
0x2104	EMU_EFPIEN_CLR	RW	EFP Interrupt Enable Register
0x3010	EMU_DECBOD_TGL	RW	DECOUPLE LVBOD Control register
0x3020	EMU_BOD3SENSE_TGL	RW	BOD3SENSE Control register
0x303C	EMU_VREGVDDCMPCTRL_TGL	RW	DC-DC VREGVDD Comparator Control Register
0x3040	EMU_PD1PARETCTRL_TGL	RW	PD1 Partial Retention Control
0x3060	EMU_LOCK_TGL	W	EMU Configuration lock register
0x3064	EMU_IF_TGL	RWH INTFLAG	Interrupt Flags
0x3068	EMU_IEN_TGL	RW	Interrupt Enables
0x306C	EMU_EM4CTRL_TGL	RW	EM4 Control
0x3070	EMU_CMD_TGL	W	EMU Command register
0x3074	EMU_CTRL_TGL	RW	EMU Control register
0x3078	EMU_TEMPLIMITS_TGL	RW SYNC	EMU Temperature thresholds
0x3084	EMU_STATUS_TGL	RH	EMU Status register
0x3088	EMU_TEMP_TGL	RH SYNC	Temperature

Offset	Name	Type	Description
0x3090	EMU_RSTCTRL_TGL	RW	Reset Management Control register
0x3094	EMU_RSTCAUSE_TGL	RH	Reset cause
0x30A0	EMU_DGIF_TGL	RWH INTFLAG	Interrupt Flags Debug
0x30A4	EMU_DGIEN_TGL	RW	Interrupt Enables Debug
0x3100	EMU_EFPIF_TGL	RWH INTFLAG	EFP Interrupt Register
0x3104	EMU_EFPIEN_TGL	RW	EFP Interrupt Enable Register

12.5 EMU Register Description

12.5.1 EMU_DECBOD - DECOUPLE LVBOD Control register

Offset	Bit Position																															
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x1	0x0			0x1	0x0
Access																											RW	RW			RW	RW
Name																											DECOVMBODMASK	DECOVMBODEN			DECBODMASK	DECBODEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	DECOVMBODMASK	0x1	RW	**Over Voltage Monitor Mask**
	DECOUPLE BOD Over Voltage Monitor Mask			
4	DECOVMBODEN	0x0	RW	**Over Voltage Monitor enable**
	DECOUPLE BOD Over Voltage Monitor enable. Enables LVBOD below vref high. BOD is masked for 20us after enable			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	DECBODMASK	0x1	RW	**DECBOD Mask**
	DECOUPLE BOD Mask			
0	DECBODEN	0x0	RW	**DECBOD enable**
	DECOUPLE BOD enable. Enables LVBOD above vref low. BOD is masked for 20us after enable			

12.5.2 EMU_BOD3SENSE - BOD3SENSE Control register

Offset	Bit Position																														
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														RW	RW	RW
Name																														VDDIO1BODEN	VDDIO0BODEN	AVDDBODEN

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	VDDIO1BODEN	0x0	RW	**VDDIO1 BOD enable**
	BOD output is automatically masked for 100us by HW after enable is set			
1	VDDIO0BODEN	0x0	RW	**VDDIO0 BOD enable**
	BOD output is automatically masked for 100us by HW after enable is set			
0	AVDDBODEN	0x0	RW	**AVDD BOD enable**
	BOD output is automatically masked for 100us by HW after enable is set			

12.5.3 EMU_VREGVDDCMPCTRL - DC-DC VREGVDD Comparator Control Register

Offset	Bit Position																														
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x3		0x0
Access																														RW		RW
Name																														THRESSEL		VREGINCMPEN

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:1	THRESSEL	0x3	RW	**VREGVDD comparator threshold programming**
	VREGVDD comparator threshold programming: 2.0->2.3V, 0.1V/step			
0	VREGINCMPEN	0x0	RW	**VREGVDD comparator enable**
	VREGVDD comparator enable. Output is masked for 5us after enabled. Automatically disabled in EM2.			

12.5.4 EMU_PD1PARETCTRL - PD1 Partial Retention Control

Offset	Bit Position																													
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									RW							
Name																									PD1PARETDIS							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	PD1PARETDIS	0x0	RW	**Disable PD1 Partial Retention**

Select PD1 register groups that are NOT retained in EM2/EM3. Each bit controls a register group. MCU core group is always retained. Bit[0]: Disables PD1 retention for MCU Peripherals group. Bit[1]: Disables PD1 retention for RADIO group. Bit [15:2]: Unused.

Value	Mode	Description
0	RETAIN	Retain associated registers when in EM2/3
1	NORETAIN	Do not retain associcated registers when in EM2/3

12.5.5 EMU_LOCK - EMU Configuration lock register

Offset	Bit Position																													
0x060	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0xADE8							
Access																									W							
Name																									LOCKKEY							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0xADE8	W	**Lock Key**

Write any other value than the unlock code to lock

Value	Mode	Description
44520	UNLOCK	Unlock EMU register

12.5.6 EMU_IF - Interrupt Flags

Offset	Bit Position																													
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0		0x0		0x0	0x0							0x0	0x0																
Access	RW	RW	RW		RW		RW	RW							RW	RW																
Name	TEMPHIGH	TEMPLOW	TEMP		TEMPAVG		VSCALEDONE	EM23WAKEUP							IOVDD0BOD	AVDDBOD																

Bit	Name	Reset	Access	Description
31	TEMPHIGH	0x0	RW	**Temperature high Interrupt flag**
	Measured temperature above threshold			
30	TEMPLOW	0x0	RW	**Temperature low Interrupt flag**
	Measured temperature below threshold			
29	TEMP	0x0	RW	**Temperature Interrupt flag**
	Temperature Update			
28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	TEMPAVG	0x0	RW	**Temperature Average Interrupt flag**
	Averaged Temperature Update			
26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	VSCALEDONE	0x0	RW	**Vscale done Interrupt flag**
	Voltage scaling completed. EM0 only.			
24	EM23WAKEUP	0x0	RW	**EM23 Wake up Interrupt flag**
	EM23 wake up			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	IOVDD0BOD	0x0	RW	**VDDIO0 BOD Interrupt flag**
	IOVDD0 BOD triggered			
16	AVDDBOD	0x0	RW	**AVDD BOD Interrupt flag**
	AVDD BOD triggered			
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

12.5.7 EMU_IEN - Interrupt Enables

Offset	Bit Position																													
0x068	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0		0x0		0x0	0x0							0x0	0x0																
Access	RW	RW	RW		RW		RW	RW							RW	RW																
Name	TEMPHIGH	TEMPLOW	TEMP		TEMPAVG		VSCALEDONE	EM23WAKEUP							IOVDD0BOD	AVDDBOD																

Bit	Name	Reset	Access	Description
31	TEMPHIGH	0x0	RW	**Temperature high Interrupt enable**
	Measured temperature above threshold Interrupt enable			
30	TEMPLOW	0x0	RW	**Temperature low Interrupt enable**
	Measured temperature below threshold Interrupt enable			
29	TEMP	0x0	RW	**Temperature Interrupt enable**
	Temperature Update Interrupt enable			
28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	TEMPAVG	0x0	RW	**Temperature Interrupt enable**
	Averaged Temperature Interrupt enable			
26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	VSCALEDONE	0x0	RW	**Vscale done Interrupt enable**
	Voltage scaling completed Interrupt enable. EM0 only.			
24	EM23WAKEUP	0x0	RW	**EM23 Wake up Interrupt enable**
	EM23 wake up Interrupt enable			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	IOVDD0BOD	0x0	RW	**VDDIO0 BOD Interrupt enable**
	IOVDD0 BOD Interrupt enable			
16	AVDDBOD	0x0	RW	**AVDD BOD Interrupt enable**
	AVDD BOD Interrupt enable			
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

12.5.8 EMU_EM4CTRL - EM4 Control

Offset	Bit Position																													
0x06C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0			0x0				0x0	
Access																								RW			RW				RW	
Name																								BOD3SENSEEM4WU			EM4IORETMODE				EM4ENTRY	

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	BOD3SENSEEM4WU	0x0	RW	**Set BOD3SENSE as EM4 wakeup**
	Enable BOD3SENSE as EM4 wakeup source			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	EM4IORETMODE	0x0	RW	**EM4 IO retention mode**
	Determine when IO retention will be applied and removed			

Value	Mode	Description
0	DISABLE	No Retention: Pads enter reset state when entering EM4
1	EM4EXIT	Retention through EM4: Pads enter reset state when exiting EM4
2	SWUNLATCH	Retention through EM4 and Wakeup: software writes UNLATCH register to remove retention

3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	EM4ENTRY	0x0	RW	**EM4 entry request**
	This field is used to enter the Energy Mode 4 sequence. Writing the sequence 2,3,2,3,2,3,2,3,2 will enter the part into Energy Mode 4			

12.5.9 EMU_CMD - EMU Command register

Offset	Bit Position																													
0x070	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0						0x0	0x0						0x0			0x0	
Access															W						W	W						W			W	
Name															RSTCAUSECLR						EM01VSCALE2	EM01VSCALE1						TEMPAVGREQ			EM4UNLATCH	

Bit	Name	Reset	Access	Description
31:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	RSTCAUSECLR	0x0	W	**Reset Cause Clear**
	Set this bit to clear the RMURSTCAUSE register			
16:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	EM01VSCALE2	0x0	W	**Scale voltage to Vscale2**
	EM01 Voltage Scale Command to scale to Voltage Scale Level 2			
10	EM01VSCALE1	0x0	W	**Scale voltage to Vscale1**
	EM01 Voltage Scale Command to scale to Voltage Scale Level 1			
9:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	TEMPAVGREQ	0x0	W	**Temperature Average Request**
	Request for Averaged Temperature Measurement			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	EM4UNLATCH	0x0	W	**EM4 unlatch**
	GPIO unlatch request after EM4 wakeup. Only valid when EM4IORETMODE== SWUNLATCH			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

12.5.10 EMU_CTRL - EMU Control register

Offset	Bit Position																														
0x074	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0													0x0							0x2						0x0			0x0
Access	RW	RW	RW													RW							RW						RW			RW
Name	EFPDRVDVDD	EFPDRVDECOUPLE	EFPDIRECTMODEEN													FLASHPWRUPONDEMAND							EM23VSCALE						TEMPAVGNUM			EM2DBGEN

Bit	Name	Reset	Access	Description
31	EFPDRVDVDD	0x0	RW	**EFP drives DVDD**
	EFP Drives DVDD			
30	EFPDRVDECOUPLE	0x0	RW	**EFP drives DECOUPLE**
	EFP Drives DECOUPLE. EMU voltage scaling is done through EFP			
29	EFPDIRECTMODEEN	0x0	RW	**EFP Direct Mode Enable**
	EFP Direct mode enable. EMU drives I2C SDA/SCL lines to transition EFP between energy modes			
28:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	FLASHPWRUPONDEMAND	0x0	RW	**Enable flash on demand wakeup**
	When set, during wake up, Flash will be in power down mode until either incoming Flash data fetch or when software issue powerup command to IMEM->MSC_CMD register			
15:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	EM23VSCALE	0x2	RW	**EM2/EM3 Vscale**
	Set VSCALE value for EM2/EM3 mode			

Value	Mode	Description
0	VSCALE0	VSCALE0. 0.9v
1	VSCALE1	VSCALE1. 1.0v
2	VSCALE2	VSCALE2. 1.1v

7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	TEMPAVGNUM	0x0	RW	**Averaged Temperature samples num**
	Number of samples taken for Averaged Temperature Measurement			

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	N16		16 measurements
	1	N64		64 measurements
2:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EM2DBGEN	0x0	RW	**Enable debugging in EM2**
	Force PD0B to stay on on EM2 entry. This allows debugger to remain connected in EM2.			

12.5.11 EMU_TEMPLIMITS - EMU Temperature thresholds

Offset	Bit Position																													
0x078	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x1FF																0x0				
Access												RW																RW				
Name												TEMPHIGH																TEMPLOW				

Bit	Name	Reset	Access	Description
31:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24:16	TEMPHIGH	0x1FF	RW	**Temp High limit**
	Temp threshold in degree Kelvin. The TEMPHIGH interrupt flag is set when a periodic temperature measurement is equal to or higher than this value.			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	TEMPLOW	0x0	RW	**Temp Low limit**
	Temp threshold in degree Kelvin. The TEMPLOW interrupt flag is set when a periodic temperature measurement is equal to or lower than this value.			

12.5.12 EMU_STATUS - EMU Status register

Offset	Bit Position																													
0x084	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																		0x0		0x0		0x0			0x2		0x0	0x0	0x0	0x0	0x0	0x0
Access																		R		R		R			R		R	R	R	R	R	R
Name																		EM2ENTERED		EM4IORET		RACACTIVE			VSCALE		VSCALEFAILED	VSCALEBUSY	TEMPAVGACTIVE	TEMPACTIVE	FIRSTTEMPDONE	LOCK

Bit	Name	Reset	Access	Description
31:15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	EM2ENTERED	0x0	R	**EM2 entered**
	Confirm chip entered EM2 state. EM2 Entry request can be delayed or denied by peripherals.			
13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	EM4IORET	0x0	R	**EM4 IO retention status**
	The status of IO retention. Will be set upon EM4 entry based on EM4IORETMODE in EMU_EM4CTRL. Cleared by setting EM4UNLATCH in EMU_CMD			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	RACACTIVE	0x0	R	**RAC active**
	This bit indicates the status of the RAC state machine. System can not enter EM2 or lower if set.			
9:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:6	VSCALE	0x2	R	**Vscale status**
	Current Voltage Scale Value			

Value	Mode	Description
0	VSCALE0	Voltage scaling set to 0.9v
1	VSCALE1	Voltage scaling set to 1.0v
2	VSCALE2	Voltage scaling set to 1.1v

5	VSCALEFAILED	0x0	R	**Vscale failed**
	Voltage scaling failed. (Time out)			
4	VSCALEBUSY	0x0	R	**Vscale busy**
	Voltage Scaling busy			
3	TEMPAVGACTIVE	0x0	R	**Temp Average active**
	Average Temperature Measurement active			
2	TEMPACTIVE	0x0	R	**Temp active**

Bit	Name	Reset	Access	Description
	Temperature Measurement active			
1	FIRSTTEMPDONE	0x0	R	**First Temp done**
	First Temperatue mesaurement completed			
0	LOCK	0x0	R	**Lock status**
	Indicates the current status of EMU Lock			

Value	Mode	Description
0	UNLOCKED	All EMU lockable registers are unlocked.
1	LOCKED	All EMU lockable registers are locked.

12.5.13 EMU_TEMP - Temperature

Offset	Bit Position																														
0x088	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x0															0x0					0x0	
Access											R															R					R	
Name											TEMPAVG															TEMP					TEMPLSB	

Bit	Name	Reset	Access	Description
31:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:16	TEMPAVG	0x0	R	**Averaged Temperature**
	Averaged Temperature Measurement. Temperature in Kelvin. 9 integer bits and 2 decimal bits (0.25 Degree resolution)			
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:2	TEMP	0x0	R	**Temperature measured**
	Temperature in Kelvin. Value of last periodic temperature measurement. Value is asynchronously updated.			
1:0	TEMPLSB	0x0	R	**Temperature measured decimal part**
	Temperature decimal part			

12.5.14 EMU_RSTCTRL - Reset Management Control register

Offset	Bit Position																													
0x090	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x1						0x1			0x0	0x0			0x0	0x1		0x1
Access																RW						RW			RW	RW			RW	RW		RW
Name																DCIRMODE						DECBODRMODE			IOVDD0BODRMODE	AVDDBODRMODE			LOCKUPRMODE	SYSRMODE		WDOG0RMODE

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	DCIRMODE	0x1	RW	**DCI System reset**
	DCI sysreset Reset Mode			
	Value	Mode		Description
	0	DISABLED		Reset request blocked
	1	ENABLED		The entire device is reset except some EMU registers
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	DECBODRMODE	0x1	RW	**Enable DECBOD reset**
	LVBOD Reset Mode. DECOUPLE monitoring. BOD must be trimmed before it is used as a reset source.			
	Value	Mode		Description
	0	DISABLED		Reset request is blocked
	1	ENABLED		The entire device is reset
9:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	IOVDD0BODRMODE	0x0	RW	**Enable VDDIO0 BOD reset**
	LEBOD2 Reset Mode. IOVDD0 monitoring. BOD must be trimmed before it is used as a reset source.			
	Value	Mode		Description
	0	DISABLED		Reset request is blocked
	1	ENABLED		The entire device is reset except some EMU registers
6	AVDDBODRMODE	0x0	RW	**Enable AVDD BOD reset**
	LEBOD1 Reset Mode. AVDD monitoring. BOD must be trimmed before it is used as a reset source.			
	Value	Mode		Description
	0	DISABLED		Reset Request is block

Bit	Name	Reset	Access	Description
	1	ENABLED		The entire device is reset except some EMU registers
5:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	LOCKUPRMODE	0x0	RW	**Enable M33 Lockup reset**
	Core LOCKUP Reset Mode			
	Value	Mode		Description
	0	DISABLED		Reset Request is Block
	1	ENABLED		The entire device is reset except some EMU registers
2	SYSRMODE	0x1	RW	**Enable M33 System reset**
	Core Sysreset Reset Mode			
	Value	Mode		Description
	0	DISABLED		Reset request is blocked
	1	ENABLED		Device is reset except some EMU registers
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	WDOG0RMODE	0x1	RW	**Enable WDOG0 reset**
	WDOG0 Reset Mode			
	Value	Mode		Description
	0	DISABLED		Reset request is blocked
	1	ENABLED		The entire device is reset except some EMU registers

12.5.15 EMU_RSTCAUSE - Reset cause

Offset	Bit Position																													
0x094	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0															0x0					0x0	0x0	0x0	0x0	0x0	0x0	0x0		0x0	0x0	0x0	0x0
Access	R															R					R	R	R	R	R	R	R		R	R	R	R
Name	VREGIN															DCI					IOVDD0BOD	AVDDBOD	DECBOD	DVDDLEBOD	DVDDBOD	SYSREQ	LOCKUP		WDOG0	EM4	PIN	POR

Bit	Name	Reset	Access	Description
31	VREGIN	0x0	R	**DCDC VREGIN comparator**
	DCDC VREGIN comparator below threshold. For Information only, not a direct source for reset. Should be used to determine whether the previous reset was caused by DCDC input being too low to support current load. In this case it is advised to keep the chip in BYPASS mode and check battery level before re-enabling integrated DCDC			
30:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	DCI	0x0	R	**DCI reset**
	Debug Challenge Interface			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	IOVDD0BOD	0x0	R	**LEBOD2 Reset**
	Brown Out Detector monitoring IOVDD0			
10	AVDDBOD	0x0	R	**LEBOD1 Reset**
	Brown Out Detector monitoring AVDD			
9	DECBOD	0x0	R	**LVBOD Reset**
	Brown Out Detector monitoring DECOUPLE			
8	DVDDLEBOD	0x0	R	**LEBOD Reset**
	Brown Out Detector monitoring DVDD in EM2/3			
7	DVDDBOD	0x0	R	**HVBOD Reset**
	Brown Out Detector monitoring DVDD in EM0/1			
6	SYSREQ	0x0	R	**M33 Core Sys Reset**
	Last Reset was as M33 Core System reset			
5	LOCKUP	0x0	R	**M33 Core Lockup Reset**
	Last Reset was as M33 Core Lockup reset			
4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	WDOG0	0x0	R	**Watchdog 0 Reset**
	Last reset was a Watchdog 0 reset			
2	EM4	0x0	R	**EM4 Wakeup Reset**
	Last reset was a EM4 Wakeup			

Bit	Name	Reset	Access	Description
1	PIN	0x0	R	**Pin Reset**
	Last reset was a Pin reset			
0	POR	0x0	R	**Power On Reset**
	Last reset was a Power On Reset			

12.5.16 EMU_DGIF - Interrupt Flags Debug

Offset	Bit Position																															
0x0A0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0					0x0																								
Access	RW	RW	RW					RW																								
Name	TEMPHIGHDGIF	TEMPLOWDGIF	TEMPDGIF					EM23WAKEUPDGIF																								

Bit	Name	Reset	Access	Description
31	TEMPHIGHDGIF	0x0	RW	**Temperature high Interrupt flag**
	Measured temperature above threshold			
30	TEMPLOWDGIF	0x0	RW	**Temperature low Interrupt flag**
	Measured temperature below threshold			
29	TEMPDGIF	0x0	RW	**Temperature Interrupt flag**
	Temperature Update			
28:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	EM23WAKEUPDGIF	0x0	RW	**EM23 Wake up Interrupt flag**
	EM23 wake up			
23:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

12.5.17 EMU_DGIEN - Interrupt Enables Debug

Offset	Bit Position																													
0x0A4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0					0x0																								
Access	RW	RW	RW					RW																								
Name	TEMPHIGHDGIEN	TEMPLOWDGIEN	TEMPDGIEN					EM23WAKEUPDGIEN																								

Bit	Name	Reset	Access	Description
31	TEMPHIGHDGIEN	0x0	RW	**Temperature high Interrupt enable**
	Measured temperature above threshold			
30	TEMPLOWDGIEN	0x0	RW	**Temperature low Interrupt enable**
	Measured temperature below threshold			
29	TEMPDGIEN	0x0	RW	**Temperature Interrupt enable**
	Temperature Update			
28:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	EM23WAKEUPDGIEN	0x0	RW	**EM23 Wake up Interrupt enable**
	EM23 wake up			
23:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

12.5.18 EMU_EFPIF - EFP Interrupt Register

Offset	Bit Position																													
0x100	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EFPIF

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EFPIF	0x0	RW	**EFP Interrupt Flag**
	EFP Interrupt			

12.5.19 EMU_EFPIEN - EFP Interrupt Enable Register

Offset	Bit Position																														
0x104	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EFPIEN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EFPIEN Enable EFP Interrupt	0x0	RW	**EFP Interrupt enable**

12.6 DCDC Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	DCDC_IPVERSION	R	IPVERSION
0x004	DCDC_EN	RW ENABLE	Enable
0x008	DCDC_CTRL	RW SYNC	Control
0x010	DCDC_EM01CTRL0	RW SYNC	EM01 Control
0x014	DCDC_EM23CTRL0	RW SYNC	EM23 Control
0x024	DCDC_IF	RWH INTFLAG	Interrupt Flags
0x028	DCDC_IEN	RW	Interrupt Enable
0x02C	DCDC_STATUS	RH	Status Register
0x040	DCDC_LOCK	W	Lock Register
0x044	DCDC_LOCKSTATUS	RH	Lock Status Register
0x1000	DCDC_IPVERSION_SET	R	IPVERSION
0x1004	DCDC_EN_SET	RW ENABLE	Enable
0x1008	DCDC_CTRL_SET	RW SYNC	Control
0x1010	DCDC_EM01CTRL0_SET	RW SYNC	EM01 Control
0x1014	DCDC_EM23CTRL0_SET	RW SYNC	EM23 Control
0x1024	DCDC_IF_SET	RWH INTFLAG	Interrupt Flags
0x1028	DCDC_IEN_SET	RW	Interrupt Enable
0x102C	DCDC_STATUS_SET	RH	Status Register
0x1040	DCDC_LOCK_SET	W	Lock Register
0x1044	DCDC_LOCKSTATUS_SET	RH	Lock Status Register
0x2000	DCDC_IPVERSION_CLR	R	IPVERSION
0x2004	DCDC_EN_CLR	RW ENABLE	Enable
0x2008	DCDC_CTRL_CLR	RW SYNC	Control
0x2010	DCDC_EM01CTRL0_CLR	RW SYNC	EM01 Control
0x2014	DCDC_EM23CTRL0_CLR	RW SYNC	EM23 Control
0x2024	DCDC_IF_CLR	RWH INTFLAG	Interrupt Flags
0x2028	DCDC_IEN_CLR	RW	Interrupt Enable
0x202C	DCDC_STATUS_CLR	RH	Status Register
0x2040	DCDC_LOCK_CLR	W	Lock Register
0x2044	DCDC_LOCKSTATUS_CLR	RH	Lock Status Register
0x3000	DCDC_IPVERSION_TGL	R	IPVERSION
0x3004	DCDC_EN_TGL	RW ENABLE	Enable
0x3008	DCDC_CTRL_TGL	RW SYNC	Control
0x3010	DCDC_EM01CTRL0_TGL	RW SYNC	EM01 Control
0x3014	DCDC_EM23CTRL0_TGL	RW SYNC	EM23 Control

Offset	Name	Type	Description
0x3024	DCDC_IF_TGL	RWH INTFLAG	Interrupt Flags
0x3028	DCDC_IEN_TGL	RW	Interrupt Enable
0x302C	DCDC_STATUS_TGL	RH	Status Register
0x3040	DCDC_LOCK_TGL	W	Lock Register
0x3044	DCDC_LOCKSTATUS_TGL	RH	Lock Status Register

12.7 DCDC Register Description

12.7.1 DCDC_IPVERSION - IPVERSION

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IPVERSION**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

12.7.2 DCDC_EN - Enable

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**Enable**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

Value	Mode	Description
0	DISABLE	Disable
1	ENABLE	Enable

12.7.3 DCDC_CTRL - Control

Offset	Bit Position																														
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																										0x4				0x1		0x0
Access																										RW				RW		RW
Name																										IPKTMAXCTRL				DCMONLYEN		MODE

Bit	Name	Reset	Access	Description
31:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	IPKTMAXCTRL	0x4	RW	**Peak Current Timeout Control**

Specifies the timeout duration when attempting to hit programmed peak current. Ton_max = (ipk_tmax_ctrl*4 + 1)*0.070us; The TMAX interrupt flag will be set if the timeout was hit before reaching peak current.

Value	Mode	Description
0	OFF	Ton_max disabled
1	TMAX_0P35US	0.35us
2	TMAX_0P63US	0.63us
3	TMAX_0P91US	0.91us
4	TMAX_1P19US	1.19us
5	TMAX_1P47US	1.47us
6	TMAX_1P75US	1.75us
7	TMAX_2P03US	2.03us

3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DCMONLYEN	0x1	RW	**DCDC DCM Only Enable**

DCDC DCM Only Enable

Value	Mode	Description
0	DUALMODE	Support higher load current at lower battery voltage by working in CCM mode
1	DCMONLYEN	DCM only mode for normal operation, this is the default setting

1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	MODE	0x0	RW	**DCDC/Bypass Mode Control**

Used to switch between bypass and dcdc regulation, this triggers a sequence of controls. IF/STATUS registers can be used to check the true status of DCDC regulator/bypass switch

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	BYPASS		DCDC is OFF, bypass switch is enabled
	1	DCDCREGULATION		Request DCDC regulation, bypass switch disabled

12.7.4 DCDC_EM01CTRL0 - EM01 Control

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x1						0x9		
Access																							RW						RW			
Name																							DRVSPEED						IPKVAL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	DRVSPEED	0x1	RW	**EM01 Drive Speed Setting**
	Used to configure drive speed for tradeoff between EMI and Efficiency			
	Value	Mode		Description
	0	BEST_EMI		Lowest Efficiency, Lowest EMI.. Small decrease in efficiency from default setting
	1	DEFAULT_SETTING		Default Efficiency, Acceptable EMI level
	2	INTERMEDIATE		Small increase in efficiency from the default setting
	3	BEST_EFFICIENCY		Highest Efficiency, Highest EMI.. Small increase in efficiency from INTERMEDIATE setting
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	IPKVAL	0x9	RW	**EM01 Peak Current Setting**
	Used to configure for required peak/load current in EM0 and EM1			
	Value	Mode		Description
	3	LOAD36MA		Ipeak = 90mA, IL = 36mA
	4	LOAD40MA		Ipeak = 100mA, IL = 40mA
	5	LOAD44MA		Ipeak = 110mA, IL = 44mA
	6	LOAD48MA		Ipeak = 120mA, IL = 48mA
	7	LOAD52MA		Ipeak = 130mA, IL = 52mA
	8	LOAD56MA		Ipeak = 140mA, IL = 56mA
	9	LOAD60MA		Ipeak = 150mA, IL = 60mA

12.7.5 DCDC_EM23CTRL0 - EM23 Control

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x1						0x3		
Access																							RW						RW		
Name																							DRVSPEED						IPKVAL		

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	DRVSPEED	0x1	RW	**EM23 Drive Speed Setting**

Used to configure drive speed for tradeoff between EMI and Efficiency

Value	Mode	Description
0	BEST_EMI	Lowest Efficiency, Lowest EMI.. Small decrease in efficiency from default setting
1	DEFAULT_SETTING	Default Efficiency, Acceptable EMI level
2	INTERMEDIATE	Small increase in efficiency from the default setting
3	BEST_EFFICIENCY	Highest Efficiency, Highest EMI.. Small increase in efficiency from INTERMEDIATE setting

7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	IPKVAL	0x3	RW	**EM23 Peak Current Setting**

Used to configure for required peak/load current in EM2 and EM3

Value	Mode	Description
3	LOAD5MA	Ipeak = 90mA, IL = 5 mA
9	LOAD10MA	Ipeak = 150mA, IL = 10 mA

12.7.6 DCDC_IF - Interrupt Flags

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																									RW	RW	RW	RW	RW	RW	RW	RW
Name																									EM4ERR	TMAX	REGULATION	VREGINHIGH	VREGINLOW	RUNNING	WARM	BYPSW

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	EM4ERR	0x0	RW	**EM4 Entry Request Error**
	EM4 entry error - software requesting EM4 entry when bypass switch is disabled			
6	TMAX	0x0	RW	**Ton_max Timeout Reached**
	Ton_max timeout was reached before peak current could be achieved			
5	REGULATION	0x0	RW	**DCDC in regulation**
	DC-DC in regulation, output voltage is in range of target voltage			
4	VREGINHIGH	0x0	RW	**VREGVDD above threshold**
	VREGVDD above threshold			
3	VREGINLOW	0x0	RW	**VREGVDD below threshold**
	VREGVDD below threshold			
2	RUNNING	0x0	RW	**DCDC Running**
	Bypass switch has turned off and DC-DC is active. Note that DC-DC might not be in regulation yet. ie output voltage may not be in range of target voltage.			
1	WARM	0x0	RW	**DCDC Warmup Time Done**
	100us DCDC warmup time complete			
0	BYPSW	0x0	RW	**Bypass Switch Enabled**
	Bypass Switch Enabled			

12.7.7 DCDC_IEN - Interrupt Enable

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																									RW	RW	RW	RW	RW	RW	RW	RW
Name																									EM4ERR	TMAX	REGULATION	VREGINHIGH	VREGINLOW	RUNNING	WARM	BYPSW

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	EM4ERR	0x0	RW	**EM4 Entry Req Interrupt Enable**
	EM4 Entry Request Error Interrupt Enable			
6	TMAX	0x0	RW	**Ton_max Timeout Interrupt Enable**
	Ton_max Timeout Interrupt Enable			
5	REGULATION	0x0	RW	**DCDC in Regulation Interrupt Enable**
	DCDC in Regulation Interrupt Enable			
4	VREGINHIGH	0x0	RW	**VREGVDD above threshold Interrupt Enable**
	VREGVDD above threshold Interrupt Enable			
3	VREGINLOW	0x0	RW	**VREGVDD below threshold Interrupt Enable**
	VREGVDD below threshold Interrupt Enable			
2	RUNNING	0x0	RW	**DCDC Running Interrupt Enable**
	DCDC Running Interrupt Enable			
1	WARM	0x0	RW	**DCDC Warmup Time Done Interrupt Enable**
	DCDC Warmup Time Done Interrupt Enable			
0	BYPSW	0x0	RW	**Bypass Switch Enabled Interrupt Enable**
	Bypass Switch Enabled Interrupt Enable			

12.7.8 DCDC_STATUS - Status Register

Offset	Bit Position																														
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												R	R	R	R	R
Name																												BYPCMPOUT	VREGIN	RUNNING	WARM	BYPSW

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	BYPCMPOUT	0x0	R	**Bypass Comparator Output**
	Bypass Comparator Output			
3	VREGIN	0x0	R	**VREGVDD comparator status**
	VREGVDD above threshold			
2	RUNNING	0x0	R	**DCDC is running**
	DCDC is running			
1	WARM	0x0	R	**DCDC Warmup Done**
	100us DCDC warmup time complete			
0	BYPSW	0x0	R	**Bypass Switch is currently enabled**
	Bypass switch is currently enabled			

12.7.9 DCDC_LOCK - Lock Register

Offset	Bit Position																															
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									LOCKKEY							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x0	W	**Configuration Lock Key**
	Write any other value than the unlock code to lock all DCDC registers			

Value	Mode	Description
43981	UNLOCKKEY	Value to write to unlock

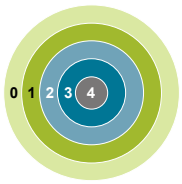
12.7.10 DCDC_LOCKSTATUS - Lock Status Register

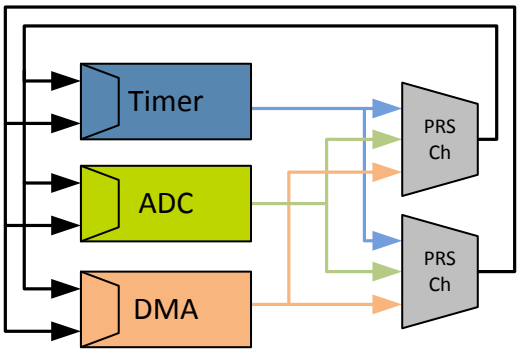
Offset	Bit Position																															
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																R
Name																																LOCK

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	LOCK	0x0	R	**Lock Status**
	Lock Status Read-Only Register			

Value	Mode	Description
0	UNLOCKED	Unlocked State
1	LOCKED	LOCKED STATE

13. PRS - Peripheral Reflex System





Quick Facts

What?

The PRS (Peripheral Reflex System) allows configurable, fast, and autonomous communication between peripherals.

Why?

Events and signals from one peripheral can be used as input signals to trigger actions in other peripherals. PRS reduces latency and ensures predictable timing by reducing software overhead and thus current consumption.

How?

Without CPU intervention the peripherals can send reflex signals to each other in single- or chained steps. The peripherals can be set up to perform actions based on the incoming reflex signals. This results in improved system performance and reduced energy consumption.

13.1 Introduction

The Peripheral Reflex System is a signal routing network allowing direct communication between different peripheral modules without involving the CPU. Peripheral modules which send out reflex signals to the PRS are called producers, and modules accepting reflex signals are called consumers. The PRS routes the reflex signals from producer to consumer peripherals, which perform actions depending on the reflex signals received.

13.2 Features

12 configurable asynchronous channels

- Each channel can be connected to any producer
- Consumers can be configured to listen to any asynchronous channel
- Can generate events to the CPU and the DMA
- Software controlled channel output using the SWPULSE and SWLEVEL registers
- Configurable logic to implement combinational functions between channels; multiple channels may be cascaded to produce more complex functions

4 configurable synchronous channels

- Special set of channels for high speed signalling between IADC and TIMER blocks

13.3 Functional Description

The PRS contains 12 asynchronous and 4 synchronous reflex channels. An overview of an asynchronous PRS reflex channel is shown in Figure 13.1 PRS Asynchronous Channel Overview on page 336. Synchronous channels are similar but do not include the configurable logic block or SWLEVEL / SWPULSE features. Asynchronous channels can be connected to any signal offered by the producers while the synchronous channels are restricted to special signals from the TIMER and IADC modules.

Similarly on the consumer side, all the peripherals can listen to asynchronous channels while only the TIMER and IADC modules can listen to synchronous channels. The consumers of a channel (synchronous or asynchronous) can choose which PRS channel to listen to and perform actions based on the reflex signals routed through that channel. Synchronous channels are only available in EM0 and EM1 while asynchronous channels are available in EM0, EM1, EM2 and EM3.

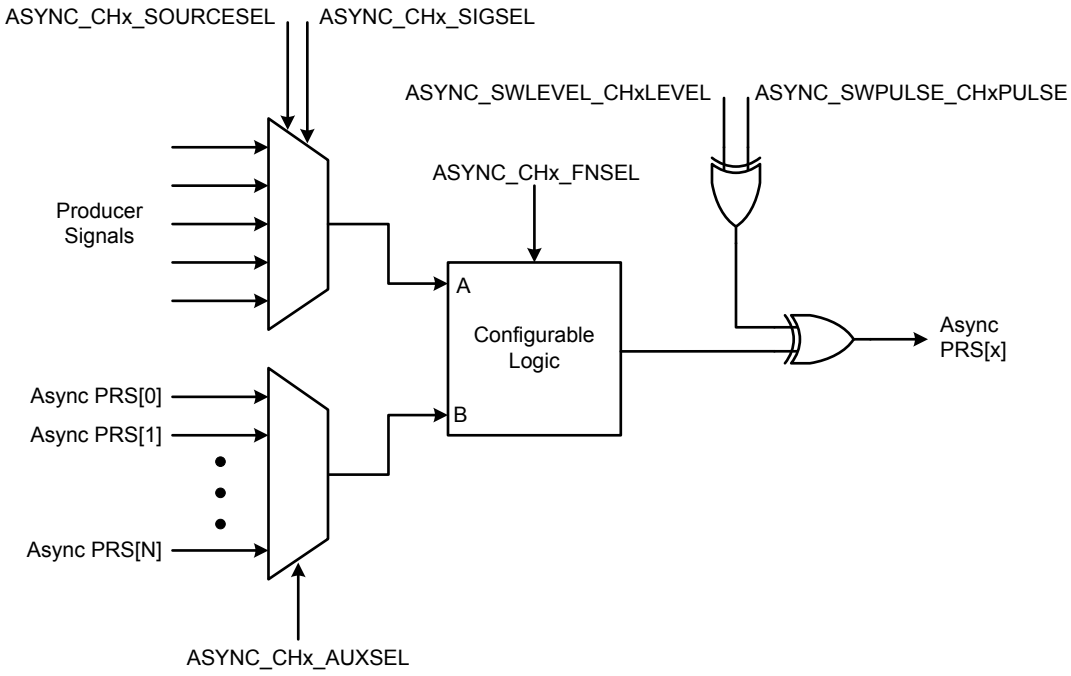


Figure 13.1. PRS Asynchronous Channel Overview

13.3.1 Asynchronous Channel Functions

Different functions can be applied to a reflex signal within the PRS. The asynchronous PRS channels can be manually triggered by writing to PRS_ASYNC_SWPULSE or PRS_ASYNC_SWLEVEL. SWLEVEL[n] is a programmable level for each asynchronous channel and holds the value it is programmed to. Setting SWPULSE[n] will cause the asynchronous channel to output a high pulse that is one EM01GRPACLK clock cycle wide. The SWLEVEL[n] and SWPULSE[n] signals are then XOR'ed with the output from the configurable logic block to form the output signal and is sent to the channel selection logic for every consumer signal. For example, when SWLEVEL[n] is set, if configurable logic produces a signal of 1, this will cause a channel output of 0.

13.3.2 Configurable Logic

The configurable logic feature enables a PRS channel to perform logic operations on the signal coming from the selected producer. Every asynchronous channel has a configurable logic block that can be programmed using the FNSEL field in the asynchronous channel control register. The configurable logic block for each channel has two inputs. Input A is the signal from the selected producer determined by SOURCESEL and SIGSEL of PRS_ASYNCHn_CTRL. Input B may be selected from the output of any other asynchronous PRS channel using the ASYNC_CHx_AUXSEL field. This allows for more complex logic functions to be created using multiple PRS channels.

Table 13.1. Configurable Logic Look up Table

A	B	FNSEL
0	0	FNSEL[0]
0	1	FNSEL[1]
1	0	FNSEL[2]
1	1	FNSEL[3]

The configurable logic feature is implemented as a 2 input look up table, with each bit of FNSEL representing the outcome for a specific input combination (see Table 13.1 Configurable Logic Look up Table on page 337). For example, if input A is 0 and input B is 1, then the PRS output will assume the value of bit 1 of FNSEL (FNSEL[1]).

To calculate the FNSEL field for an "A NAND B" function, the truth table can be filled out as:

Table 13.2. A NAND B Example

A	B	FNSEL = (A NAND B)
0	0	FNSEL[0] = 1
0	1	FNSEL[1] = 1
1	0	FNSEL[2] = 1
1	1	FNSEL[3] = 0

In this example, the value of FNSEL has been calculated to be 0111 (binary), or 0x7.

Using the FNSEL field, a total of 16 two-input logic functions can be implemented, as shown in Table 13.3 List of Logic Functions on page 337.

Table 13.3. List of Logic Functions

FNSEL value	Implemented Function
0x0	0
0x1	A NOR B
0x2	(NOT A) AND B
0x3	NOT A
0x4	A AND (NOT B)
0x5	(NOT B)
0x6	A XOR B
0x7	A NAND B
0x8	A AND B
0x9	A XNOR B

FNSEL value	Implemented Function
0xA	B
0xB	A OR (NOT B)
0xC	A
0xD	(NOT A) OR B
0xE	A OR B
0xF	1

The default value of FNSEL is 0xC, meaning that the input from the selected producer goes through unchanged. This feature can be used to combine multiple channels to get even more complex functions.

13.3.3 Producers

Through SOURCESEL in PRS_SYNCHx_CTRL or PRS_ASYNCHx_CTRL, each PRS channel (synchronous and asynchronous respectively) selects its signal producers. Each producer outputs one or more signals which can be selected by setting the SIGSEL field. Setting the SOURCESEL bits to 0 (Off) leads to a constant 0 output from the input mux regardless of SIGSEL.

The GPIO producer signals depend on settings in the GPIO module. They are selected using the edge interrupt configuration settings described in 25.3.10.1 Edge Interrupt Generation. PIN0 uses settings for the EXTI0 interrupt, PIN1 uses settings for EXTI1, and so on.

For example, to route PB00 as a producer for PRS channel 2, EXTI0, EXTI1, EXTI2, or EXTI3 should be configured to connect to PB00, and the corresponding GPIO PINx should be selected as the PRS channel 2 producer. If we choose EXTI1 via PRS producer "GPIO PIN1":

1. GPIO_EXTIPSELL_EXTIPSEL1 = PORTB, and GPIO_EXTIPINSELL_EXTIPINSEL1 = PIN0 connect PB00 through the EXTI1 signal.
2. PRS_ASYNC_CH2_CTRL_SOURCESEL = GPIO, and PRS_ASYNC_CH2_CTRL_SIGSEL = PIN1 connects the PIN1 (EXTI1) signal to asynchornous PRS channel 2 as a producer.

13.3.3.1 Producer Details

Table 13.4. Synchronous PRS Producers

Peripheral	SOURCESEL	Signal	SIGSEL
TIMER0	TIMER0 (0x01)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4
TIMER1	TIMER1 (0x02)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4
IADC0	IADC0 (0x03)	SCANENTRYDONE	0x0
		SCANTABLEDONE	0x1
		SINGLEDONE	0x2
TIMER2	TIMER2 (0x04)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4
TIMER3	TIMER3 (0x05)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4
TIMER4	TIMER4 (0x06)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4

Table 13.5. Asynchronous PRS Producers

Peripheral	SOURCESEL	Signal	SIGSEL
IADC0	IADC0 (0x01)	SCANENTRYDONE	0x0
		SCANTABLEDONE	0x1
		SINGLEDONE	0x2
LETIMER0	LETIMER0 (0x02)	CH0	0x0
		CH1	0x1
RTCC	RTCC (0x03)	CCV0	0x0
		CCV1	0x1
		CCV2	0x2
BURTC	BURTC (0x04)	COMP	0x0
		OVERFLOW	0x1
GPIO	GPIO (0x05)	PIN0	0x0
		PIN1	0x1
		PIN2	0x2
		PIN3	0x3
		PIN4	0x4
		PIN5	0x5
		PIN6	0x6
		PIN7	0x7
CMU	CMUL (0x06)	CLKOUT0	0x0
		CLKOUT1	0x1
		CLKOUT2	0x2
PRS	PRSL (0x0A)	ASYNCH0	0x0
		ASYNCH1	0x1
		ASYNCH2	0x2
		ASYNCH3	0x3
		ASYNCH4	0x4
		ASYNCH5	0x5
		ASYNCH6	0x6
		ASYNCH7	0x7
	PRS (0x0B)	ASYNCH8	0x0
		ASYNCH9	0x1
		ASYNCH10	0x2
		ASYNCH11	0x3

Peripheral	SOURCESEL	Signal	SIGSEL
EUART0	EUART0 (0x0C)	IRDATX	0x0
		RTS	0x1
		TX	0x2
		TXC	0x3
		RXFL	0x4
USART0	USART0 (0x20)	CS	0x0
		IRTX	0x1
		RTS	0x2
		RXDATA	0x3
		TX	0x4
		TXC	0x5
USART1	USART1 (0x21)	CS	0x0
		IRTX	0x1
		RTS	0x2
		RXDATA	0x3
		TX	0x4
		TXC	0x5
TIMER0	TIMER0 (0x22)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4
TIMER1	TIMER1 (0x23)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4
TIMER2	TIMER2 (0x24)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4
TIMER3	TIMER3 (0x25)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4

Peripheral	SOURCESEL	Signal	SIGSEL
CORE	CORE (0x26)	CTIOUT0	0x0
		CTIOUT1	0x1
		CTIOUT2	0x2
		CTIOUT3	0x3
AGC	AGCL (0x27)	CCA	0x0
		CCAREQ	0x1
	AGC (0x28)	RSSIDONE	0x2
MODEM	MODEML (0x2A)	ADVANCE	0x0
		ANT0	0x1
		ANT1	0x2
		COHDSADET	0x3
		COHDSALIVE	0x4
		DCLK	0x5
		DOUT	0x6
		FRAMEDET	0x7
	MODEM (0x2B)	FRAMESENT	0x0
		LOWCORR	0x1
		LRDSADET	0x2
		LRDSALIVE	0x3
		NEWSYMBOL	0x4
		NEWWND	0x5
		POSTPONE	0x6
		PREDET	0x7
	MODEMH (0x2C)	PRESENT	0x0
		RSSIJUMP	0x1
		SYNCSENT	0x2
		TIMDET	0x3
		WEAK	0x4
		EOF	0x5
FRC	FRC (0x2D)	DCLK	0x0
		DOUT	0x1
PROTIMER	PROTIMERL (0x2E)	LBTF	0x6
		LBTR	0x7
	PROTIMER (0x2F)	LBTS	0x0
PDM	PDML (0x31)	PDMDSRPULSE	0x0

Peripheral	SOURCESEL	Signal	SIGSEL
RAC	RACL (0x33)	ACTIVE	0x0
		LNAEN	0x1
		PAEN	0x2
		RX	0x3
		TX	0x4
		CTIOUT0	0x5
		CTIOUT1	0x6
		CTIOUT2	0x7
	RAC (0x34)	CTIOUT3	0x0
TIMER4	TIMER4 (0x35)	UF	0x0
		OF	0x1
		CC0	0x2
		CC1	0x3
		CC2	0x4
LFRCO	LFRCO (0x38)	CALMEAS	0x0
		SDM	0x1
		TCMEAS	0x2

13.3.4 Consumers

Consumer peripherals can be set to listen to a PRS channel and perform an action based on the signal received on that channel. This is done by programming the PRSSEL or SPRSSEL in the consumer registers. SPRSSEL is only present for signals with the ability to listen to synchronous channels. The consumer registers follow the naming convention PRS_CONSUMER_<peripheral_name>_<signal_name>. For example, the PRS_CONSUMER_TIMER0_CC0 register is used to select which PRS channel output is sent to the TIMER0 peripheral's CC0 signal. In turn, the target peripheral should be configured to use the associated PRS trigger as desired. This is described in the individual peripheral chapters.

Note: When configuring the synchronous PRS consumer registers, the target peripheral should be disabled or configured to not use the affected PRS signal. This will ensure that no false triggers occur at the consumer.

13.3.4.1 Event on PRS

The PRS can be used to send events to the MCU to wake the system. This is very useful in combination with the Wait For Event (WFE) instruction. Any asynchronous PRS channel can be selected for this using PRSSEL in PRS_CONSUMER_CORE_M33RXEV.

Using this feature, one can e.g. set up a timer to trigger an event to the MCU periodically, every time letting the MCU continue from a WFE instruction in its program. This can help in performance-critical sections where timing is known, and the goal is to wait for an event, execute some code, then wait for another event, execute some code, and so on.

13.3.4.2 DMA Request on PRS

Up to two independent DMA requests can be generated by the PRS. The PRS asynchronous channels triggering the DMA requests are selected with the PRSSEL fields in the PRS_CONSUMER_LDMAXBAR_DMAREQx registers.The requests are set whenever the selected asynchronous PRS outputs are high.

13.4 PRS Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	PRS_IPVERSION	R	IP version ID
0x008	PRS_ASYNC_SWPULSE	W	Software Pulse Register
0x00C	PRS_ASYNC_SWLEVEL	RW	Software Level Register
0x010	PRS_ASYNC_PEEK	RH	Async Channel Values
0x014	PRS_SYNC_PEEK	RH	Sync Channel Values
0x018	PRS_ASYNC_CHx_CTRL	RW	Async Channel Control Register
0x048	PRS_SYNC_CHx_CTRL	RW	Sync Channel Control Register
0x058	PRS_CONSUM-ER_CMU_CALDN	RW	CMU CALDN Consumer Selection
0x05C	PRS_CONSUMER_CMU_CAL-UP	RW	CMU CALUP Consumer Selection
0x064	PRS_CONSUM-ER_IADC0_SCANTRIGGER	RW	IADC0 SCANTRIGGER Consumer Selection
0x068	PRS_CONSUMER_IADC0_SIN-GLETRIGGER	RW	IADC0 SINGLETRIGGER Consumer Selection
0x06C	PRS_CONSUMER_LDMAX-BAR_DMAREQ0	RW	DMAREQ0 Consumer Selection
0x070	PRS_CONSUMER_LDMAX-BAR_DMAREQ1	RW	DMAREQ1 Consumer Selection
0x074	PRS_CONSUMER_LETIM-ER0_CLEAR	RW	LETIMER CLEAR Consumer Selection
0x078	PRS_CONSUMER_LETIM-ER0_START	RW	LETIMER START Consumer Selection
0x07C	PRS_CONSUMER_LETIM-ER0_STOP	RW	LETIMER STOP Consumer Selection
0x080	PRS_CONSUMER_EUART0_RX	RW	EUART0 RX consumer register
0x084	PRS_CONSUMER_EU-ART0_TRIGGER	RW	EUART0 TRIGGER Consumer register
0x088	PRS_CONSUMER_MO-DEM_DIN	RW	MODEM DIN Consumer Selection
0x0C0	PRS_CONSUMER_RAC_CLR	RW	RAC CLR Consumer Selection
0x0C4	PRS_CONSUMER_RAC_CTIIN0	RW	RAC CTIIN0 Consumer Selection
0x0C8	PRS_CONSUMER_RAC_CTIIN1	RW	RAC CTIIN1 Consumer Selection
0x0CC	PRS_CONSUMER_RAC_CTIIN2	RW	RAC CTIIN2 Consumer Selection
0x0D0	PRS_CONSUMER_RAC_CTIIN3	RW	RAC CTIIN3 Consumer Selection
0x0D4	PRS_CONSUM-ER_RAC_FORCETX	RW	RAC FORCETX Consumer Selection
0x0D8	PRS_CONSUMER_RAC_RXDIS	RW	RAC RXDIS Consumer Selection
0x0DC	PRS_CONSUMER_RAC_RXEN	RW	RAC RXEN Consumer Selection
0x0E0	PRS_CONSUMER_RAC_SEQ	RW	RAC SEQ Consumer Selection

Offset	Name	Type	Description
0x0E4	PRS_CONSUMER_RAC_TXEN	RW	RAC TXEN Consumer Selection
0x0E8	PRS_CONSUMER_RTCC_CC0	RW	RTCC CC0 Consumer Selection
0x0EC	PRS_CONSUMER_RTCC_CC1	RW	RTCC CC1 Consumer Selection
0x0F0	PRS_CONSUMER_RTCC_CC2	RW	RTCC CC2 Consumer Selection
0x0F8	PRS_CONSUM-ER_CORE_CTIIN0	RW	CTI0 Consumer Selection
0x0FC	PRS_CONSUM-ER_CORE_CTIIN1	RW	CTI1 Consumer Selection
0x100	PRS_CONSUM-ER_CORE_CTIIN2	RW	CTI2 Consumer Selection
0x104	PRS_CONSUM-ER_CORE_CTIIN3	RW	CTI3 Consumer Selection
0x108	PRS_CONSUM-ER_CORE_M33RXEV	RW	M33 Consumer Selection
0x10C	PRS_CONSUMER_TIM-ER0_CC0	RW	TIMER0 CC0 Consumer Selection
0x110	PRS_CONSUMER_TIM-ER0_CC1	RW	TIMER0 CC1 Consumer Selection
0x114	PRS_CONSUMER_TIM-ER0_CC2	RW	TIMER0 CC2 Consumer Selection
0x118	PRS_CONSUMER_TIMER0_DTI	RW	TIMER0 DTI Consumer Selection
0x11C	PRS_CONSUMER_TIM-ER0_DTIFS1	RW	TIMER0 DTIFS1 Consumer Selection
0x120	PRS_CONSUMER_TIM-ER0_DTIFS2	RW	TIMER0 DTIFS2 Consumer Selection
0x124	PRS_CONSUMER_TIM-ER1_CC0	RW	TIMER1 CC0 Consumer Selection
0x128	PRS_CONSUMER_TIM-ER1_CC1	RW	TIMER1 CC1 Consumer Selection
0x12C	PRS_CONSUMER_TIM-ER1_CC2	RW	TIMER1 CC2 Consumer Selection
0x130	PRS_CONSUMER_TIMER1_DTI	RW	TIMER1 DTI Consumer Selection
0x134	PRS_CONSUMER_TIM-ER1_DTIFS1	RW	TIMER1 DTIFS1 Consumer Selection
0x138	PRS_CONSUMER_TIM-ER1_DTIFS2	RW	TIMER1 DTIFS2 Consumer Selection
0x13C	PRS_CONSUMER_TIM-ER2_CC0	RW	TIMER2 CC0 Consumer Selection
0x140	PRS_CONSUMER_TIM-ER2_CC1	RW	TIMER2 CC1 Consumer Selection
0x144	PRS_CONSUMER_TIM-ER2_CC2	RW	TIMER2 CC2 Consumer Selection
0x148	PRS_CONSUMER_TIMER2_DTI	RW	TIMER2 DTI Consumer Selection
0x14C	PRS_CONSUMER_TIM-ER2_DTIFS1	RW	TIMER2 DTIFS1 Consumer Selection

Offset	Name	Type	Description
0x150	PRS_CONSUMER_TIMER2_DTIFS2	RW	TIMER2 DTIFS2 Consumer Selection
0x154	PRS_CONSUMER_TIMER3_CC0	RW	TIMER3 CC0 Consumer Selection
0x158	PRS_CONSUMER_TIMER3_CC1	RW	TIMER3 CC1 Consumer Selection
0x15C	PRS_CONSUMER_TIMER3_CC2	RW	TIMER3 CC2 Consumer Selection
0x160	PRS_CONSUMER_TIMER3_DTI	RW	TIMER3 DTI Consumer Selection
0x164	PRS_CONSUMER_TIMER3_DTIFS1	RW	TIMER3 DTIFS1 Consumer Selection
0x168	PRS_CONSUMER_TIMER3_DTIFS2	RW	TIMER3 DTIFS2 Consumer Selection
0x16C	PRS_CONSUMER_TIMER4_CC0	RW	TIMER4 CC0 Consumer Selection
0x170	PRS_CONSUMER_TIMER4_CC1	RW	TIMER4 CC1 Consumer Selection
0x174	PRS_CONSUMER_TIMER4_CC2	RW	TIMER4 CC2 Consumer Selection
0x178	PRS_CONSUMER_TIMER4_DTI	RW	TIMER4 DTI Consumer Selection
0x17C	PRS_CONSUMER_TIMER4_DTIFS1	RW	TIMER4 DTIFS1 Consumer Selection
0x180	PRS_CONSUMER_TIMER4_DTIFS2	RW	TIMER4 DTIFS2 Consumer Selection
0x184	PRS_CONSUMER_USART0_CLK	RW	USART0 CLK Consumer Selection
0x188	PRS_CONSUMER_USART0_IR	RW	USART0 IR Consumer Selection
0x18C	PRS_CONSUMER_USART0_RX	RW	USART0 RX Consumer Selection
0x190	PRS_CONSUMER_USART0_TRIGGER	RW	USART0 TRIGGER Consumer Selection
0x194	PRS_CONSUMER_USART1_CLK	RW	USART1 CLK Consumer Selection
0x198	PRS_CONSUMER_USART1_IR	RW	USART1 IR Consumer Selection
0x19C	PRS_CONSUMER_USART1_RX	RW	USART1 RX Consumer Selection
0x1A0	PRS_CONSUMER_USART1_TRIGGER	RW	USART1 TRIGGER Consumer Selection
0x1A4	PRS_CONSUMER_WDOG0_SRC0	RW	WDOG0 SRC0 Consumer Selection
0x1A8	PRS_CONSUMER_WDOG0_SRC1	RW	WDOG0 SRC1 Consumer Selection
0x1000	PRS_IPVERSION_SET	R	IP version ID
0x1008	PRS_ASYNC_SWPULSE_SET	W	Software Pulse Register
0x100C	PRS_ASYNC_SWLEVEL_SET	RW	Software Level Register
0x1010	PRS_ASYNC_PEEK_SET	RH	Async Channel Values

Offset	Name	Type	Description
0x1014	PRS_SYNC_PEEK_SET	RH	Sync Channel Values
0x1018	PRS_ASYNC_CHx_CTRL_SET	RW	Async Channel Control Register
0x1048	PRS_SYNC_CHx_CTRL_SET	RW	Sync Channel Control Register
0x1058	PRS_CONSUMER_CMU_CALDN_SET	RW	CMU CALDN Consumer Selection
0x105C	PRS_CONSUMER_CMU_CALUP_SET	RW	CMU CALUP Consumer Selection
0x1064	PRS_CONSUMER_IADC0_SCANTRIGGER_SET	RW	IADC0 SCANTRIGGER Consumer Selection
0x1068	PRS_CONSUMER_IADC0_SINGLETRIGGER_SET	RW	IADC0 SINGLETRIGGER Consumer Selection
0x106C	PRS_CONSUMER_LDMAXBAR_DMAREQ0_SET	RW	DMAREQ0 Consumer Selection
0x1070	PRS_CONSUMER_LDMAXBAR_DMAREQ1_SET	RW	DMAREQ1 Consumer Selection
0x1074	PRS_CONSUMER_LETIMER0_CLEAR_SET	RW	LETIMER CLEAR Consumer Selection
0x1078	PRS_CONSUMER_LETIMER0_START_SET	RW	LETIMER START Consumer Selection
0x107C	PRS_CONSUMER_LETIMER0_STOP_SET	RW	LETIMER STOP Consumer Selection
0x1080	PRS_CONSUMER_EUART0_RX_SET	RW	EUART0 RX consumer register
0x1084	PRS_CONSUMER_EUART0_TRIGGER_SET	RW	EUART0 TRIGGER Consumer register
0x1088	PRS_CONSUMER_MODEM_DIN_SET	RW	MODEM DIN Consumer Selection
0x10C0	PRS_CONSUMER_RAC_CLR_SET	RW	RAC CLR Consumer Selection
0x10C4	PRS_CONSUMER_RAC_CTIIN0_SET	RW	RAC CTIIN0 Consumer Selection
0x10C8	PRS_CONSUMER_RAC_CTIIN1_SET	RW	RAC CTIIN1 Consumer Selection
0x10CC	PRS_CONSUMER_RAC_CTIIN2_SET	RW	RAC CTIIN2 Consumer Selection
0x10D0	PRS_CONSUMER_RAC_CTIIN3_SET	RW	RAC CTIIN3 Consumer Selection
0x10D4	PRS_CONSUMER_RAC_FORCETX_SET	RW	RAC FORCETX Consumer Selection
0x10D8	PRS_CONSUMER_RAC_RXDIS_SET	RW	RAC RXDIS Consumer Selection
0x10DC	PRS_CONSUMER_RAC_RXEN_SET	RW	RAC RXEN Consumer Selection
0x10E0	PRS_CONSUMER_RAC_SEQ_SET	RW	RAC SEQ Consumer Selection

Offset	Name	Type	Description
0x10E4	PRS_CONSUMER_RAC_TXEN_SET	RW	RAC TXEN Consumer Selection
0x10E8	PRS_CONSUMER_RTCC_CC0_SET	RW	RTCC CC0 Consumer Selection
0x10EC	PRS_CONSUMER_RTCC_CC1_SET	RW	RTCC CC1 Consumer Selection
0x10F0	PRS_CONSUMER_RTCC_CC2_SET	RW	RTCC CC2 Consumer Selection
0x10F8	PRS_CONSUMER_CORE_CTIIN0_SET	RW	CTI0 Consumer Selection
0x10FC	PRS_CONSUMER_CORE_CTIIN1_SET	RW	CTI1 Consumer Selection
0x1100	PRS_CONSUMER_CORE_CTIIN2_SET	RW	CTI2 Consumer Selection
0x1104	PRS_CONSUMER_CORE_CTIIN3_SET	RW	CTI3 Consumer Selection
0x1108	PRS_CONSUMER_CORE_M33RXEV_SET	RW	M33 Consumer Selection
0x110C	PRS_CONSUMER_TIMER0_CC0_SET	RW	TIMER0 CC0 Consumer Selection
0x1110	PRS_CONSUMER_TIMER0_CC1_SET	RW	TIMER0 CC1 Consumer Selection
0x1114	PRS_CONSUMER_TIMER0_CC2_SET	RW	TIMER0 CC2 Consumer Selection
0x1118	PRS_CONSUMER_TIMER0_DTI_SET	RW	TIMER0 DTI Consumer Selection
0x111C	PRS_CONSUMER_TIMER0_DTIFS1_SET	RW	TIMER0 DTIFS1 Consumer Selection
0x1120	PRS_CONSUMER_TIMER0_DTIFS2_SET	RW	TIMER0 DTIFS2 Consumer Selection
0x1124	PRS_CONSUMER_TIMER1_CC0_SET	RW	TIMER1 CC0 Consumer Selection
0x1128	PRS_CONSUMER_TIMER1_CC1_SET	RW	TIMER1 CC1 Consumer Selection
0x112C	PRS_CONSUMER_TIMER1_CC2_SET	RW	TIMER1 CC2 Consumer Selection
0x1130	PRS_CONSUMER_TIMER1_DTI_SET	RW	TIMER1 DTI Consumer Selection
0x1134	PRS_CONSUMER_TIMER1_DTIFS1_SET	RW	TIMER1 DTIFS1 Consumer Selection
0x1138	PRS_CONSUMER_TIMER1_DTIFS2_SET	RW	TIMER1 DTIFS2 Consumer Selection
0x113C	PRS_CONSUMER_TIMER2_CC0_SET	RW	TIMER2 CC0 Consumer Selection
0x1140	PRS_CONSUMER_TIMER2_CC1_SET	RW	TIMER2 CC1 Consumer Selection

Offset	Name	Type	Description
0x1144	PRS_CONSUMER_TIMER2_CC2_SET	RW	TIMER2 CC2 Consumer Selection
0x1148	PRS_CONSUMER_TIMER2_DTI_SET	RW	TIMER2 DTI Consumer Selection
0x114C	PRS_CONSUMER_TIMER2_DTIFS1_SET	RW	TIMER2 DTIFS1 Consumer Selection
0x1150	PRS_CONSUMER_TIMER2_DTIFS2_SET	RW	TIMER2 DTIFS2 Consumer Selection
0x1154	PRS_CONSUMER_TIMER3_CC0_SET	RW	TIMER3 CC0 Consumer Selection
0x1158	PRS_CONSUMER_TIMER3_CC1_SET	RW	TIMER3 CC1 Consumer Selection
0x115C	PRS_CONSUMER_TIMER3_CC2_SET	RW	TIMER3 CC2 Consumer Selection
0x1160	PRS_CONSUMER_TIMER3_DTI_SET	RW	TIMER3 DTI Consumer Selection
0x1164	PRS_CONSUMER_TIMER3_DTIFS1_SET	RW	TIMER3 DTIFS1 Consumer Selection
0x1168	PRS_CONSUMER_TIMER3_DTIFS2_SET	RW	TIMER3 DTIFS2 Consumer Selection
0x116C	PRS_CONSUMER_TIMER4_CC0_SET	RW	TIMER4 CC0 Consumer Selection
0x1170	PRS_CONSUMER_TIMER4_CC1_SET	RW	TIMER4 CC1 Consumer Selection
0x1174	PRS_CONSUMER_TIMER4_CC2_SET	RW	TIMER4 CC2 Consumer Selection
0x1178	PRS_CONSUMER_TIMER4_DTI_SET	RW	TIMER4 DTI Consumer Selection
0x117C	PRS_CONSUMER_TIMER4_DTIFS1_SET	RW	TIMER4 DTIFS1 Consumer Selection
0x1180	PRS_CONSUMER_TIMER4_DTIFS2_SET	RW	TIMER4 DTIFS2 Consumer Selection
0x1184	PRS_CONSUMER_USART0_CLK_SET	RW	USART0 CLK Consumer Selection
0x1188	PRS_CONSUMER_USART0_IR_SET	RW	USART0 IR Consumer Selection
0x118C	PRS_CONSUMER_USART0_RX_SET	RW	USART0 RX Consumer Selection
0x1190	PRS_CONSUMER_USART0_TRIGGER_SET	RW	USART0 TRIGGER Consumer Selection
0x1194	PRS_CONSUMER_USART1_CLK_SET	RW	USART1 CLK Consumer Selection
0x1198	PRS_CONSUMER_USART1_IR_SET	RW	USART1 IR Consumer Selection
0x119C	PRS_CONSUMER_USART1_RX_SET	RW	USART1 RX Consumer Selection

Offset	Name	Type	Description
0x11A0	PRS_CONSUMER_USART1_TRIGGER_SET	RW	USART1 TRIGGER Consumer Selection
0x11A4	PRS_CONSUMER_WDOG0_SRC0_SET	RW	WDOG0 SRC0 Consumer Selection
0x11A8	PRS_CONSUMER_WDOG0_SRC1_SET	RW	WDOG0 SRC1 Consumer Selection
0x2000	PRS_IPVERSION_CLR	R	IP version ID
0x2008	PRS_ASYNC_SWPULSE_CLR	W	Software Pulse Register
0x200C	PRS_ASYNC_SWLEVEL_CLR	RW	Software Level Register
0x2010	PRS_ASYNC_PEEK_CLR	RH	Async Channel Values
0x2014	PRS_SYNC_PEEK_CLR	RH	Sync Channel Values
0x2018	PRS_ASYNC_CHx_CTRL_CLR	RW	Async Channel Control Register
0x2048	PRS_SYNC_CHx_CTRL_CLR	RW	Sync Channel Control Register
0x2058	PRS_CONSUMER_CMU_CALDN_CLR	RW	CMU CALDN Consumer Selection
0x205C	PRS_CONSUMER_CMU_CALUP_CLR	RW	CMU CALUP Consumer Selection
0x2064	PRS_CONSUMER_IADC0_SCANTRIGGER_CLR	RW	IADC0 SCANTRIGGER Consumer Selection
0x2068	PRS_CONSUMER_IADC0_SINGLETRIGGER_CLR	RW	IADC0 SINGLETRIGGER Consumer Selection
0x206C	PRS_CONSUMER_LDMAXBAR_DMAREQ0_CLR	RW	DMAREQ0 Consumer Selection
0x2070	PRS_CONSUMER_LDMAXBAR_DMAREQ1_CLR	RW	DMAREQ1 Consumer Selection
0x2074	PRS_CONSUMER_LETIMER0_CLEAR_CLR	RW	LETIMER CLEAR Consumer Selection
0x2078	PRS_CONSUMER_LETIMER0_START_CLR	RW	LETIMER START Consumer Selection
0x207C	PRS_CONSUMER_LETIMER0_STOP_CLR	RW	LETIMER STOP Consumer Selection
0x2080	PRS_CONSUMER_EUART0_RX_CLR	RW	EUART0 RX consumer register
0x2084	PRS_CONSUMER_EUART0_TRIGGER_CLR	RW	EUART0 TRIGGER Consumer register
0x2088	PRS_CONSUMER_MODEM_DIN_CLR	RW	MODEM DIN Consumer Selection
0x20C0	PRS_CONSUMER_RAC_CLR_CLR	RW	RAC CLR Consumer Selection
0x20C4	PRS_CONSUMER_RAC_CTIIN0_CLR	RW	RAC CTIIN0 Consumer Selection
0x20C8	PRS_CONSUMER_RAC_CTIIN1_CLR	RW	RAC CTIIN1 Consumer Selection

Offset	Name	Type	Description
0x20CC	PRS_CONSUMER_RAC_CTIIN2_CLR	RW	RAC CTIIN2 Consumer Selection
0x20D0	PRS_CONSUMER_RAC_CTIIN3_CLR	RW	RAC CTIIN3 Consumer Selection
0x20D4	PRS_CONSUMER_RAC_FORCETX_CLR	RW	RAC FORCETX Consumer Selection
0x20D8	PRS_CONSUMER_RAC_RXDIS_CLR	RW	RAC RXDIS Consumer Selection
0x20DC	PRS_CONSUMER_RAC_RXEN_CLR	RW	RAC RXEN Consumer Selection
0x20E0	PRS_CONSUMER_RAC_SEQ_CLR	RW	RAC SEQ Consumer Selection
0x20E4	PRS_CONSUMER_RAC_TXEN_CLR	RW	RAC TXEN Consumer Selection
0x20E8	PRS_CONSUMER_RTCC_CC0_CLR	RW	RTCC CC0 Consumer Selection
0x20EC	PRS_CONSUMER_RTCC_CC1_CLR	RW	RTCC CC1 Consumer Selection
0x20F0	PRS_CONSUMER_RTCC_CC2_CLR	RW	RTCC CC2 Consumer Selection
0x20F8	PRS_CONSUMER_CORE_CTIIN0_CLR	RW	CTI0 Consumer Selection
0x20FC	PRS_CONSUMER_CORE_CTIIN1_CLR	RW	CTI1 Consumer Selection
0x2100	PRS_CONSUMER_CORE_CTIIN2_CLR	RW	CTI2 Consumer Selection
0x2104	PRS_CONSUMER_CORE_CTIIN3_CLR	RW	CTI3 Consumer Selection
0x2108	PRS_CONSUMER_CORE_M33RXEV_CLR	RW	M33 Consumer Selection
0x210C	PRS_CONSUMER_TIMER0_CC0_CLR	RW	TIMER0 CC0 Consumer Selection
0x2110	PRS_CONSUMER_TIMER0_CC1_CLR	RW	TIMER0 CC1 Consumer Selection
0x2114	PRS_CONSUMER_TIMER0_CC2_CLR	RW	TIMER0 CC2 Consumer Selection
0x2118	PRS_CONSUMER_TIMER0_DTI_CLR	RW	TIMER0 DTI Consumer Selection
0x211C	PRS_CONSUMER_TIMER0_DTIFS1_CLR	RW	TIMER0 DTIFS1 Consumer Selection
0x2120	PRS_CONSUMER_TIMER0_DTIFS2_CLR	RW	TIMER0 DTIFS2 Consumer Selection
0x2124	PRS_CONSUMER_TIMER1_CC0_CLR	RW	TIMER1 CC0 Consumer Selection
0x2128	PRS_CONSUMER_TIMER1_CC1_CLR	RW	TIMER1 CC1 Consumer Selection

Offset	Name	Type	Description
0x212C	PRS_CONSUMER_TIMER1_CC2_CLR	RW	TIMER1 CC2 Consumer Selection
0x2130	PRS_CONSUMER_TIMER1_DTI_CLR	RW	TIMER1 DTI Consumer Selection
0x2134	PRS_CONSUMER_TIMER1_DTIFS1_CLR	RW	TIMER1 DTIFS1 Consumer Selection
0x2138	PRS_CONSUMER_TIMER1_DTIFS2_CLR	RW	TIMER1 DTIFS2 Consumer Selection
0x213C	PRS_CONSUMER_TIMER2_CC0_CLR	RW	TIMER2 CC0 Consumer Selection
0x2140	PRS_CONSUMER_TIMER2_CC1_CLR	RW	TIMER2 CC1 Consumer Selection
0x2144	PRS_CONSUMER_TIMER2_CC2_CLR	RW	TIMER2 CC2 Consumer Selection
0x2148	PRS_CONSUMER_TIMER2_DTI_CLR	RW	TIMER2 DTI Consumer Selection
0x214C	PRS_CONSUMER_TIMER2_DTIFS1_CLR	RW	TIMER2 DTIFS1 Consumer Selection
0x2150	PRS_CONSUMER_TIMER2_DTIFS2_CLR	RW	TIMER2 DTIFS2 Consumer Selection
0x2154	PRS_CONSUMER_TIMER3_CC0_CLR	RW	TIMER3 CC0 Consumer Selection
0x2158	PRS_CONSUMER_TIMER3_CC1_CLR	RW	TIMER3 CC1 Consumer Selection
0x215C	PRS_CONSUMER_TIMER3_CC2_CLR	RW	TIMER3 CC2 Consumer Selection
0x2160	PRS_CONSUMER_TIMER3_DTI_CLR	RW	TIMER3 DTI Consumer Selection
0x2164	PRS_CONSUMER_TIMER3_DTIFS1_CLR	RW	TIMER3 DTIFS1 Consumer Selection
0x2168	PRS_CONSUMER_TIMER3_DTIFS2_CLR	RW	TIMER3 DTIFS2 Consumer Selection
0x216C	PRS_CONSUMER_TIMER4_CC0_CLR	RW	TIMER4 CC0 Consumer Selection
0x2170	PRS_CONSUMER_TIMER4_CC1_CLR	RW	TIMER4 CC1 Consumer Selection
0x2174	PRS_CONSUMER_TIMER4_CC2_CLR	RW	TIMER4 CC2 Consumer Selection
0x2178	PRS_CONSUMER_TIMER4_DTI_CLR	RW	TIMER4 DTI Consumer Selection
0x217C	PRS_CONSUMER_TIMER4_DTIFS1_CLR	RW	TIMER4 DTIFS1 Consumer Selection
0x2180	PRS_CONSUMER_TIMER4_DTIFS2_CLR	RW	TIMER4 DTIFS2 Consumer Selection
0x2184	PRS_CONSUMER_USART0_CLK_CLR	RW	USART0 CLK Consumer Selection

Offset	Name	Type	Description
0x2188	PRS_CONSUMER_USART0_IR_CLR	RW	USART0 IR Consumer Selection
0x218C	PRS_CONSUMER_USART0_RX_CLR	RW	USART0 RX Consumer Selection
0x2190	PRS_CONSUMER_USART0_TRIGGER_CLR	RW	USART0 TRIGGER Consumer Selection
0x2194	PRS_CONSUMER_USART1_CLK_CLR	RW	USART1 CLK Consumer Selection
0x2198	PRS_CONSUMER_USART1_IR_CLR	RW	USART1 IR Consumer Selection
0x219C	PRS_CONSUMER_USART1_RX_CLR	RW	USART1 RX Consumer Selection
0x21A0	PRS_CONSUMER_USART1_TRIGGER_CLR	RW	USART1 TRIGGER Consumer Selection
0x21A4	PRS_CONSUMER_WDOG0_SRC0_CLR	RW	WDOG0 SRC0 Consumer Selection
0x21A8	PRS_CONSUMER_WDOG0_SRC1_CLR	RW	WDOG0 SRC1 Consumer Selection
0x3000	PRS_IPVERSION_TGL	R	IP version ID
0x3008	PRS_ASYNC_SWPULSE_TGL	W	Software Pulse Register
0x300C	PRS_ASYNC_SWLEVEL_TGL	RW	Software Level Register
0x3010	PRS_ASYNC_PEEK_TGL	RH	Async Channel Values
0x3014	PRS_SYNC_PEEK_TGL	RH	Sync Channel Values
0x3018	PRS_ASYNC_CHx_CTRL_TGL	RW	Async Channel Control Register
0x3048	PRS_SYNC_CHx_CTRL_TGL	RW	Sync Channel Control Register
0x3058	PRS_CONSUMER_CMU_CALDN_TGL	RW	CMU CALDN Consumer Selection
0x305C	PRS_CONSUMER_CMU_CALUP_TGL	RW	CMU CALUP Consumer Selection
0x3064	PRS_CONSUMER_IADC0_SCANTRIGGER_TGL	RW	IADC0 SCANTRIGGER Consumer Selection
0x3068	PRS_CONSUMER_IADC0_SINGLETRIGGER_TGL	RW	IADC0 SINGLETRIGGER Consumer Selection
0x306C	PRS_CONSUMER_LDMAXBAR_DMAREQ0_TGL	RW	DMAREQ0 Consumer Selection
0x3070	PRS_CONSUMER_LDMAXBAR_DMAREQ1_TGL	RW	DMAREQ1 Consumer Selection
0x3074	PRS_CONSUMER_LETIMER0_CLEAR_TGL	RW	LETIMER CLEAR Consumer Selection
0x3078	PRS_CONSUMER_LETIMER0_START_TGL	RW	LETIMER START Consumer Selection
0x307C	PRS_CONSUMER_LETIMER0_STOP_TGL	RW	LETIMER STOP Consumer Selection

Offset	Name	Type	Description
0x3080	PRS_CONSUMER_EUART0_RX_TGL	RW	EUART0 RX consumer register
0x3084	PRS_CONSUMER_EUART0_TRIGGER_TGL	RW	EUART0 TRIGGER Consumer register
0x3088	PRS_CONSUMER_MODEM_DIN_TGL	RW	MODEM DIN Consumer Selection
0x30C0	PRS_CONSUMER_RAC_CLR_TGL	RW	RAC CLR Consumer Selection
0x30C4	PRS_CONSUMER_RAC_CTIIN0_TGL	RW	RAC CTIIN0 Consumer Selection
0x30C8	PRS_CONSUMER_RAC_CTIIN1_TGL	RW	RAC CTIIN1 Consumer Selection
0x30CC	PRS_CONSUMER_RAC_CTIIN2_TGL	RW	RAC CTIIN2 Consumer Selection
0x30D0	PRS_CONSUMER_RAC_CTIIN3_TGL	RW	RAC CTIIN3 Consumer Selection
0x30D4	PRS_CONSUMER_RAC_FORCETX_TGL	RW	RAC FORCETX Consumer Selection
0x30D8	PRS_CONSUMER_RAC_RXDIS_TGL	RW	RAC RXDIS Consumer Selection
0x30DC	PRS_CONSUMER_RAC_RXEN_TGL	RW	RAC RXEN Consumer Selection
0x30E0	PRS_CONSUMER_RAC_SEQ_TGL	RW	RAC SEQ Consumer Selection
0x30E4	PRS_CONSUMER_RAC_TXEN_TGL	RW	RAC TXEN Consumer Selection
0x30E8	PRS_CONSUMER_RTCC_CC0_TGL	RW	RTCC CC0 Consumer Selection
0x30EC	PRS_CONSUMER_RTCC_CC1_TGL	RW	RTCC CC1 Consumer Selection
0x30F0	PRS_CONSUMER_RTCC_CC2_TGL	RW	RTCC CC2 Consumer Selection
0x30F8	PRS_CONSUMER_CORE_CTIIN0_TGL	RW	CTI0 Consumer Selection
0x30FC	PRS_CONSUMER_CORE_CTIIN1_TGL	RW	CTI1 Consumer Selection
0x3100	PRS_CONSUMER_CORE_CTIIN2_TGL	RW	CTI2 Consumer Selection
0x3104	PRS_CONSUMER_CORE_CTIIN3_TGL	RW	CTI3 Consumer Selection
0x3108	PRS_CONSUMER_CORE_M33RXEV_TGL	RW	M33 Consumer Selection
0x310C	PRS_CONSUMER_TIMER0_CC0_TGL	RW	TIMER0 CC0 Consumer Selection
0x3110	PRS_CONSUMER_TIMER0_CC1_TGL	RW	TIMER0 CC1 Consumer Selection

Offset	Name	Type	Description
0x3114	PRS_CONSUMER_TIMER0_CC2_TGL	RW	TIMER0 CC2 Consumer Selection
0x3118	PRS_CONSUMER_TIMER0_DTI_TGL	RW	TIMER0 DTI Consumer Selection
0x311C	PRS_CONSUMER_TIMER0_DTIFS1_TGL	RW	TIMER0 DTIFS1 Consumer Selection
0x3120	PRS_CONSUMER_TIMER0_DTIFS2_TGL	RW	TIMER0 DTIFS2 Consumer Selection
0x3124	PRS_CONSUMER_TIMER1_CC0_TGL	RW	TIMER1 CC0 Consumer Selection
0x3128	PRS_CONSUMER_TIMER1_CC1_TGL	RW	TIMER1 CC1 Consumer Selection
0x312C	PRS_CONSUMER_TIMER1_CC2_TGL	RW	TIMER1 CC2 Consumer Selection
0x3130	PRS_CONSUMER_TIMER1_DTI_TGL	RW	TIMER1 DTI Consumer Selection
0x3134	PRS_CONSUMER_TIMER1_DTIFS1_TGL	RW	TIMER1 DTIFS1 Consumer Selection
0x3138	PRS_CONSUMER_TIMER1_DTIFS2_TGL	RW	TIMER1 DTIFS2 Consumer Selection
0x313C	PRS_CONSUMER_TIMER2_CC0_TGL	RW	TIMER2 CC0 Consumer Selection
0x3140	PRS_CONSUMER_TIMER2_CC1_TGL	RW	TIMER2 CC1 Consumer Selection
0x3144	PRS_CONSUMER_TIMER2_CC2_TGL	RW	TIMER2 CC2 Consumer Selection
0x3148	PRS_CONSUMER_TIMER2_DTI_TGL	RW	TIMER2 DTI Consumer Selection
0x314C	PRS_CONSUMER_TIMER2_DTIFS1_TGL	RW	TIMER2 DTIFS1 Consumer Selection
0x3150	PRS_CONSUMER_TIMER2_DTIFS2_TGL	RW	TIMER2 DTIFS2 Consumer Selection
0x3154	PRS_CONSUMER_TIMER3_CC0_TGL	RW	TIMER3 CC0 Consumer Selection
0x3158	PRS_CONSUMER_TIMER3_CC1_TGL	RW	TIMER3 CC1 Consumer Selection
0x315C	PRS_CONSUMER_TIMER3_CC2_TGL	RW	TIMER3 CC2 Consumer Selection
0x3160	PRS_CONSUMER_TIMER3_DTI_TGL	RW	TIMER3 DTI Consumer Selection
0x3164	PRS_CONSUMER_TIMER3_DTIFS1_TGL	RW	TIMER3 DTIFS1 Consumer Selection
0x3168	PRS_CONSUMER_TIMER3_DTIFS2_TGL	RW	TIMER3 DTIFS2 Consumer Selection
0x316C	PRS_CONSUMER_TIMER4_CC0_TGL	RW	TIMER4 CC0 Consumer Selection

Offset	Name	Type	Description
0x3170	PRS_CONSUMER_TIMER4_CC1_TGL	RW	TIMER4 CC1 Consumer Selection
0x3174	PRS_CONSUMER_TIMER4_CC2_TGL	RW	TIMER4 CC2 Consumer Selection
0x3178	PRS_CONSUMER_TIMER4_DTI_TGL	RW	TIMER4 DTI Consumer Selection
0x317C	PRS_CONSUMER_TIMER4_DTIFS1_TGL	RW	TIMER4 DTIFS1 Consumer Selection
0x3180	PRS_CONSUMER_TIMER4_DTIFS2_TGL	RW	TIMER4 DTIFS2 Consumer Selection
0x3184	PRS_CONSUMER_USART0_CLK_TGL	RW	USART0 CLK Consumer Selection
0x3188	PRS_CONSUMER_USART0_IR_TGL	RW	USART0 IR Consumer Selection
0x318C	PRS_CONSUMER_USART0_RX_TGL	RW	USART0 RX Consumer Selection
0x3190	PRS_CONSUMER_USART0_TRIGGER_TGL	RW	USART0 TRIGGER Consumer Selection
0x3194	PRS_CONSUMER_USART1_CLK_TGL	RW	USART1 CLK Consumer Selection
0x3198	PRS_CONSUMER_USART1_IR_TGL	RW	USART1 IR Consumer Selection
0x319C	PRS_CONSUMER_USART1_RX_TGL	RW	USART1 RX Consumer Selection
0x31A0	PRS_CONSUMER_USART1_TRIGGER_TGL	RW	USART1 TRIGGER Consumer Selection
0x31A4	PRS_CONSUMER_WDOG0_SRC0_TGL	RW	WDOG0 SRC0 Consumer Selection
0x31A8	PRS_CONSUMER_WDOG0_SRC1_TGL	RW	WDOG0 SRC1 Consumer Selection

13.5 PRS Register Description

13.5.1 PRS_IPVERSION - IP version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x1	R	**New BitField**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

13.5.2 PRS_ASYNC_SWPULSE - Software Pulse Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																					W	W	W	W	W	W	W	W	W	W	W	W
Name																					CH11PULSE	CH10PULSE	CH9PULSE	CH8PULSE	CH7PULSE	CH6PULSE	CH5PULSE	CH4PULSE	CH3PULSE	CH2PULSE	CH1PULSE	CH0PULSE

Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	CH11PULSE	0x0	W	**Channel pulse**
	Channel 11 pulse generation			
10	CH10PULSE	0x0	W	**Channel pulse**
	Channel 10 pulse generation			
9	CH9PULSE	0x0	W	**Channel pulse**
	Channel 9 pulse generation			
8	CH8PULSE	0x0	W	**Channel pulse**
	Channel 8 pulse generation			
7	CH7PULSE	0x0	W	**Channel pulse**
	Channel 7 pulse generation			
6	CH6PULSE	0x0	W	**Channel pulse**
	Channel 6 pulse generation			
5	CH5PULSE	0x0	W	**Channel pulse**
	Channel 5 pulse generation			
4	CH4PULSE	0x0	W	**Channel pulse**
	Channel 4 pulse generation			
3	CH3PULSE	0x0	W	**Channel pulse**
	Channel 3 pulse generation			
2	CH2PULSE	0x0	W	**Channel pulse**
	Channel 2 pulse generation			
1	CH1PULSE	0x0	W	**Channel pulse**
	Channel 1 pulse generation			
0	CH0PULSE	0x0	W	**Channel pulse**
	Channel 0 pulse generation			

13.5.3 PRS_ASYNC_SWLEVEL - Software Level Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																					CH11LEVEL	CH10LEVEL	CH9LEVEL	CH8LEVEL	CH7LEVEL	CH6LEVEL	CH5LEVEL	CH4LEVEL	CH3LEVEL	CH2LEVEL	CH1LEVEL	CH0LEVEL

Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	CH11LEVEL	0x0	RW	**Channel Level**
	Channel 11 Software Level			
10	CH10LEVEL	0x0	RW	**Channel Level**
	Channel 10 Software Level			
9	CH9LEVEL	0x0	RW	**Channel Level**
	Channel 9 Software Level			
8	CH8LEVEL	0x0	RW	**Channel Level**
	Channel 8 Software Level			
7	CH7LEVEL	0x0	RW	**Channel Level**
	Channel 7 Software Level			
6	CH6LEVEL	0x0	RW	**Channel Level**
	Channel 6 Software Level			
5	CH5LEVEL	0x0	RW	**Channel Level**
	Channel 5 Software Level			
4	CH4LEVEL	0x0	RW	**Channel Level**
	Channel 4 Software Level			
3	CH3LEVEL	0x0	RW	**Channel Level**
	Channel 3 Software Level			
2	CH2LEVEL	0x0	RW	**Channel Level**
	Channel 2 Software Level			
1	CH1LEVEL	0x0	RW	**Channel Level**
	Channel 1 Software Level			
0	CH0LEVEL	0x0	RW	**Channel Level**
	Channel 0 Software Level			

13.5.4 PRS_ASYNC_PEEK - Async Channel Values

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																					R	R	R	R	R	R	R	R	R	R	R	R
Name																					CH11VAL	CH10VAL	CH9VAL	CH8VAL	CH7VAL	CH6VAL	CH5VAL	CH4VAL	CH3VAL	CH2VAL	CH1VAL	CH0VAL

Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	CH11VAL	0x0	R	**Channel 11 Current Value**
	See bit 0.			
10	CH10VAL	0x0	R	**Channel 10 Current Value**
	See bit 0.			
9	CH9VAL	0x0	R	**Channel 9 Current Value**
	See bit 0.			
8	CH8VAL	0x0	R	**Channel 8 Current Value**
	See bit 0.			
7	CH7VAL	0x0	R	**Channel 7 Current Value**
	See bit 0.			
6	CH6VAL	0x0	R	**Channel 6 Current Value**
	See bit 0.			
5	CH5VAL	0x0	R	**Channel 5 Current Value**
	See bit 0.			
4	CH4VAL	0x0	R	**Channel 4 Current Value**
	See bit 0.			
3	CH3VAL	0x0	R	**Channel 3 Current Value**
	See bit 0.			
2	CH2VAL	0x0	R	**Channel 2 Current Value**
	See bit 0.			
1	CH1VAL	0x0	R	**Channel 1 Current Value**
	See bit 0.			
0	CH0VAL	0x0	R	**Channel 0 Current Value**
	Sample the current output value of channel 0. This value may be one or two clock delayed			

13.5.5 PRS_SYNC_PEEK - Sync Channel Values

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0	0x0	0x0
Access																													R	R	R	R
Name																													CH3VAL	CH2VAL	CH1VAL	CH0VAL

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	CH3VAL	0x0	R	**Channel Value**
	Channel 3 current value			
2	CH2VAL	0x0	R	**Channel Value**
	Channel 2 current value			
1	CH1VAL	0x0	R	**Channel Value**
	Channel 1 current value			
0	CH0VAL	0x0	R	**Channel Value**
	Channel 0 current value			

13.5.6 PRS_ASYNC_CHx_CTRL - Async Channel Control Register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0								0xC					0x0												0x0		
Access					RW								RW					RW												RW		
Name					AUXSEL								FNSEL					SOURCESEL												SIGSEL		

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:24	AUXSEL	0x0	RW	**Auxiliary LUT Input Select**
	Select Asynchronous PRS channel as input B of LUT function. Async PRS[n] is selected with AUXSEL = n.			
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	FNSEL	0xC	RW	**Function Select**
	Look up Table function select			

Value	Mode	Description
0	LOGICAL_ZERO	Logical 0
1	A_NOR_B	A NOR B
2	NOT_A_AND_B	(!A) AND B
3	NOT_A	!A
4	A_AND_NOT_B	A AND (!B)
5	NOT_B	!B
6	A_XOR_B	A XOR B
7	A_NAND_B	A NAND B
8	A_AND_B	A AND B
9	A_XNOR_B	A XNOR B
10	B	B
11	NOT_A_OR_B	(!A) OR B
12	A	A
13	A_OR_NOT_B	A OR (!B)
14	A_OR_B	A OR B
15	LOGICAL_ONE	Logical 1

15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	SOURCESEL	0x0	RW	**Source Select**

Bit	Name	Reset	Access	Description
	Select input source to async PRS channel.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	SIGSEL	0x0	RW	**Signal Select**
	Select signal input to async PRS channel.			

Value	Mode	Description
0	NONE	

13.5.7 PRS_SYNC_CHx_CTRL - Sync Channel Control Register

Offset	Bit Position																													
0x048	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0x0										0x0	
Access																					RW										RW	
Name																					SOURCESEL										SIGSEL	

Bit	Name	Reset	Access	Description
31:15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	SOURCESEL	0x0	RW	**Source Select**
	Select input source to sync PRS channel.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	SIGSEL	0x0	RW	**Signal Select**
	Select signal input to sync PRS channel.			

13.5.8 PRS_CONSUMER_CMU_CALDN - CMU CALDN Consumer Selection

Offset	Bit Position																															
0x058	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CALDN async channel select**
	CALDN async channel select			

13.5.9 PRS_CONSUMER_CMU_CALUP - CMU CALUP Consumer Selection

Offset	Bit Position																															
0x05C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CALUP async channel select**
	CALUP async channel select			

13.5.10 PRS_CONSUMER_IADC0_SCANTRIGGER - IADC0 SCANTRIGGER Consumer Selection

Offset	Bit Position																													
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**SCAN sync channel select**
	SCAN sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**SCAN async channel select**
	SCAN async channel select			

13.5.11 PRS_CONSUMER_IADC0_SINGLETRIGGER - IADC0 SINGLETRIGGER Consumer Selection

Offset	Bit Position																													
0x068	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**SINGLE sync channel select**
	SINGLE sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**SINGLE async channel select**
	SINGLE async channel select			

13.5.12 PRS_CONSUMER_LDMAXBAR_DMAREQ0 - DMAREQ0 Consumer Selection

Offset	Bit Position																															
0x06C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DMAREQ0 async channel select**
	DMAREQ0 async channel select			

13.5.13 PRS_CONSUMER_LDMAXBAR_DMAREQ1 - DMAREQ1 Consumer Selection

Offset	Bit Position																															
0x070	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DMAREQ1 async channel select**
	DMAREQ1 async channel select			

13.5.14 PRS_CONSUMER_LETIMER0_CLEAR - LETIMER CLEAR Consumer Selection

Offset	Bit Position																															
0x074	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CLEAR async channel select**
	CLEAR async channel select			

13.5.15 PRS_CONSUMER_LETIMER0_START - LETIMER START Consumer Selection

Offset	Bit Position																															
0x078	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**START async channel select**
	START async channel select			

13.5.16 PRS_CONSUMER_LETIMER0_STOP - LETIMER STOP Consumer Selection

Offset	Bit Position																															
0x07C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**STOP async channel select**
	STOP async channel select			

13.5.17 PRS_CONSUMER_EUART0_RX - EUART0 RX consumer register

Offset	Bit Position																															
0x080	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**RX async channel select**
	RX async channel select			

13.5.18 PRS_CONSUMER_EUART0_TRIGGER - EUART0 TRIGGER Consumer register

Offset	Bit Position																															
0x084	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													RW		
Name																													PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**TRIGGER async channel select**
	TRIGGER async channel select			

13.5.19 PRS_CONSUMER_MODEM_DIN - MODEM DIN Consumer Selection

Offset	Bit Position																															
0x088	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													RW		
Name																													PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DIN async channel select**
	DIN async channel select			

13.5.20 PRS_CONSUMER_RAC_CLR - RAC CLR Consumer Selection

Offset	Bit Position																															
0x0C0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CLR async channel select**
	CLR async channel select			

13.5.21 PRS_CONSUMER_RAC_CTIIN0 - RAC CTIIN0 Consumer Selection

Offset	Bit Position																															
0x0C4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CTI async channel select**
	CTI async channel select			

13.5.22 PRS_CONSUMER_RAC_CTIIN1 - RAC CTIIN1 Consumer Selection

Offset	Bit Position																															
0x0C8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CTI async channel select**
	CTI async channel select			

13.5.23 PRS_CONSUMER_RAC_CTIIN2 - RAC CTIIN2 Consumer Selection

Offset	Bit Position																															
0x0CC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CTI async channel select**
	CTI async channel select			

13.5.24 PRS_CONSUMER_RAC_CTIIN3 - RAC CTIIN3 Consumer Selection

Offset	Bit Position																															
0x0D0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CTI async channel select**
	CTI async channel select			

13.5.25 PRS_CONSUMER_RAC_FORCETX - RAC FORCETX Consumer Selection

Offset	Bit Position																															
0x0D4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**FORCETX async channel select**
	FORCETX async channel select			

13.5.26 PRS_CONSUMER_RAC_RXDIS - RAC RXDIS Consumer Selection

Offset	Bit Position																															
0x0D8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**RXDIS async channel select**
	RXDIS async channel select			

13.5.27 PRS_CONSUMER_RAC_RXEN - RAC RXEN Consumer Selection

Offset	Bit Position																															
0x0DC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**RXEN async channel select**
	RXEN async channel select			

13.5.28 PRS_CONSUMER_RAC_SEQ - RAC SEQ Consumer Selection

Offset	Bit Position																															
0x0E0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**SEQ async channel select**
	SEQ async channel select			

13.5.29 PRS_CONSUMER_RAC_TXEN - RAC TXEN Consumer Selection

Offset	Bit Position																															
0x0E4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**TXEN async channel select**
	TXEN async channel select			

13.5.30 PRS_CONSUMER_RTCC_CC0 - RTCC CC0 Consumer Selection

Offset	Bit Position																															
0x0E8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC0 async channel select**
	CC0 async channel select			

13.5.31 PRS_CONSUMER_RTCC_CC1 - RTCC CC1 Consumer Selection

Offset	Bit Position																															
0x0EC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC1 async channel select**
	CC1 async channel select			

13.5.32 PRS_CONSUMER_RTCC_CC2 - RTCC CC2 Consumer Selection

Offset	Bit Position																															
0x0F0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC2 async channel select**
	CC2 async channel select			

13.5.33 PRS_CONSUMER_CORE_CTIIN0 - CTI0 Consumer Selection

Offset	Bit Position																															
0x0F8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CTI async channel select**
	CTI async channel select			

13.5.34 PRS_CONSUMER_CORE_CTIIN1 - CTI1 Consumer Selection

Offset	Bit Position																															
0x0FC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CTI async channel select**
	CTI async channel select			

13.5.35 PRS_CONSUMER_CORE_CTIIN2 - CTI2 Consumer Selection

Offset	Bit Position																															
0x100	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CTI async channel select**
	CTI async channel select			

13.5.36 PRS_CONSUMER_CORE_CTIIN3 - CTI3 Consumer Selection

Offset	Bit Position																															
0x104	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CTI async channel select**
	CTI async channel select			

13.5.37 PRS_CONSUMER_CORE_M33RXEV - M33 Consumer Selection

Offset	Bit Position																															
0x108	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**M33 async channel select**
	M33 async channel select			

13.5.38 PRS_CONSUMER_TIMER0_CC0 - TIMER0 CC0 Consumer Selection

Offset	Bit Position																													
0x10C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC0 sync channel select**
	CC0 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC0 async channel select**
	CC0 async channel select			

13.5.39 PRS_CONSUMER_TIMER0_CC1 - TIMER0 CC1 Consumer Selection

Offset	Bit Position																													
0x110	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC1 sync channel select**
	CC1 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC1 async channel select**
	CC1 async channel select			

13.5.40 PRS_CONSUMER_TIMER0_CC2 - TIMER0 CC2 Consumer Selection

Offset	Bit Position																															
0x114	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC2 sync channel select**
	CC2 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC2 async channel select**
	CC2 async channel select			

13.5.41 PRS_CONSUMER_TIMER0_DTI - TIMER0 DTI Consumer Selection

Offset	Bit Position																															
0x118	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0		
Access																														RW		
Name																														PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.42 PRS_CONSUMER_TIMER0_DTIFS1 - TIMER0 DTIFS1 Consumer Selection

Offset	Bit Position																															
0x11C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.43 PRS_CONSUMER_TIMER0_DTIFS2 - TIMER0 DTIFS2 Consumer Selection

Offset	Bit Position																															
0x120	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.44 PRS_CONSUMER_TIMER1_CC0 - TIMER1 CC0 Consumer Selection

Offset	Bit Position																													
0x124	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC0 sync channel select**
	CC0 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC0 async channel select**
	CC0 async channel select			

13.5.45 PRS_CONSUMER_TIMER1_CC1 - TIMER1 CC1 Consumer Selection

Offset	Bit Position																													
0x128	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC1 sync channel select**
	CC1 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC1 async channel select**
	CC1 async channel select			

13.5.46 PRS_CONSUMER_TIMER1_CC2 - TIMER1 CC2 Consumer Selection

Offset	Bit Position																															
0x12C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC2 sync channel select**
	CC2 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC2 async channel select**
	CC2 async channel select			

13.5.47 PRS_CONSUMER_TIMER1_DTI - TIMER1 DTI Consumer Selection

Offset	Bit Position																															
0x130	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0		
Access																														RW		
Name																														PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.48 PRS_CONSUMER_TIMER1_DTIFS1 - TIMER1 DTIFS1 Consumer Selection

Offset	Bit Position																															
0x134	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.49 PRS_CONSUMER_TIMER1_DTIFS2 - TIMER1 DTIFS2 Consumer Selection

Offset	Bit Position																															
0x138	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.50 PRS_CONSUMER_TIMER2_CC0 - TIMER2 CC0 Consumer Selection

Offset	Bit Position																													
0x13C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC0 sync channel select**
	CC0 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC0 async channel select**
	CC0 async channel select			

13.5.51 PRS_CONSUMER_TIMER2_CC1 - TIMER2 CC1 Consumer Selection

Offset	Bit Position																													
0x140	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC1 sync channel select**
	CC1 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC1 async channel select**
	CC1 async channel select			

13.5.52 PRS_CONSUMER_TIMER2_CC2 - TIMER2 CC2 Consumer Selection

Offset	Bit Position																													
0x144	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0		
Access																							RW						RW		
Name																							SPRSSEL						PRSSEL		

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC2 sync channel select**
	CC2 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC2 async channel select**
	CC2 async channel select			

13.5.53 PRS_CONSUMER_TIMER2_DTI - TIMER2 DTI Consumer Selection

Offset	Bit Position																													
0x148	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													RW		
Name																													PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.54 PRS_CONSUMER_TIMER2_DTIFS1 - TIMER2 DTIFS1 Consumer Selection

Offset	Bit Position																															
0x14C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.55 PRS_CONSUMER_TIMER2_DTIFS2 - TIMER2 DTIFS2 Consumer Selection

Offset	Bit Position																															
0x150	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.56 PRS_CONSUMER_TIMER3_CC0 - TIMER3 CC0 Consumer Selection

Offset	Bit Position																													
0x154	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC0 sync channel select**
	CC0 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC0 async channel select**
	CC0 async channel select			

13.5.57 PRS_CONSUMER_TIMER3_CC1 - TIMER3 CC1 Consumer Selection

Offset	Bit Position																													
0x158	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC1 sync channel select**
	CC1 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC1 async channel select**
	CC1 async channel select			

13.5.58 PRS_CONSUMER_TIMER3_CC2 - TIMER3 CC2 Consumer Selection

Offset	Bit Position																													
0x15C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC2 sync channel select**
	CC2 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC2 async channel select**
	CC2 async channel select			

13.5.59 PRS_CONSUMER_TIMER3_DTI - TIMER3 DTI Consumer Selection

Offset	Bit Position																													
0x160	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													RW		
Name																													PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.60 PRS_CONSUMER_TIMER3_DTIFS1 - TIMER3 DTIFS1 Consumer Selection

Offset	Bit Position																															
0x164	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.61 PRS_CONSUMER_TIMER3_DTIFS2 - TIMER3 DTIFS2 Consumer Selection

Offset	Bit Position																															
0x168	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.62 PRS_CONSUMER_TIMER4_CC0 - TIMER4 CC0 Consumer Selection

Offset	Bit Position																													
0x16C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC0 sync channel select**
	CC0 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC0 async channel select**
	CC0 async channel select			

13.5.63 PRS_CONSUMER_TIMER4_CC1 - TIMER4 CC1 Consumer Selection

Offset	Bit Position																													
0x170	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC1 sync channel select**
	CC1 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC1 async channel select**
	CC1 async channel select			

13.5.64 PRS_CONSUMER_TIMER4_CC2 - TIMER4 CC2 Consumer Selection

Offset	Bit Position																															
0x174	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0						0x0			
Access																							RW						RW			
Name																							SPRSSEL						PRSSEL			

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	SPRSSEL	0x0	RW	**CC2 sync channel select**
	CC2 sync channel select			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CC2 async channel select**
	CC2 async channel select			

13.5.65 PRS_CONSUMER_TIMER4_DTI - TIMER4 DTI Consumer Selection

Offset	Bit Position																															
0x178	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0		
Access																														RW		
Name																														PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.66 PRS_CONSUMER_TIMER4_DTIFS1 - TIMER4 DTIFS1 Consumer Selection

Offset	Bit Position																															
0x17C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.67 PRS_CONSUMER_TIMER4_DTIFS2 - TIMER4 DTIFS2 Consumer Selection

Offset	Bit Position																															
0x180	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**DTI async channel select**
	DTI async channel select			

13.5.68 PRS_CONSUMER_USART0_CLK - USART0 CLK Consumer Selection

Offset	Bit Position																															
0x184	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CLK async channel select**
	CLK async channel select			

13.5.69 PRS_CONSUMER_USART0_IR - USART0 IR Consumer Selection

Offset	Bit Position																															
0x188	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**IR async channel select**
	IR async channel select			

13.5.70 PRS_CONSUMER_USART0_RX - USART0 RX Consumer Selection

Offset	Bit Position																															
0x18C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**RX async channel select**
	RX async channel select			

13.5.71 PRS_CONSUMER_USART0_TRIGGER - USART0 TRIGGER Consumer Selection

Offset	Bit Position																															
0x190	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**TRIGGER async channel select**
	TRIGGER async channel select			

13.5.72 PRS_CONSUMER_USART1_CLK - USART1 CLK Consumer Selection

Offset	Bit Position																															
0x194	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**CLK async channel select**
	CLK async channel select			

13.5.73 PRS_CONSUMER_USART1_IR - USART1 IR Consumer Selection

Offset	Bit Position																															
0x198	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													PRSSEL			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**IR async channel select**
	IR async channel select			

13.5.74 PRS_CONSUMER_USART1_RX - USART1 RX Consumer Selection

Offset	Bit Position																															
0x19C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													RW		
Name																													PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**RX async channel select**
	RX async channel select			

13.5.75 PRS_CONSUMER_USART1_TRIGGER - USART1 TRIGGER Consumer Selection

Offset	Bit Position																															
0x1A0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													RW		
Name																													PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**TRIGGER async channel select**
	TRIGGER async channel select			

13.5.76 PRS_CONSUMER_WDOG0_SRC0 - WDOG0 SRC0 Consumer Selection

Offset	Bit Position																															
0x1A4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													RW		
Name																													PRSSEL		

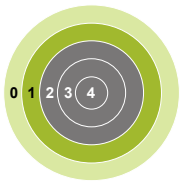
Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**SRC0 async channel select**
	SRC0 async channel select			

13.5.77 PRS_CONSUMER_WDOG0_SRC1 - WDOG0 SRC1 Consumer Selection

Offset	Bit Position																															
0x1A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													RW		
Name																													PRSSEL		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSEL	0x0	RW	**SRC1 async channel select**
	SRC1 async channel select			

14. GPCRC - General Purpose Cyclic Redundancy Check



Quick Facts

What?

The GPCRC is an error-detecting module commonly used in digital networks and storage systems to detect accidental changes to data.

Why?

The GPCRC module can detect errors in data, giving a higher system reliability and robustness.

How?

Blocks of data entering GPCRC module can have a short checksum, based on the remainder of a polynomial division of their contents; on retrieval the calculation is repeated, and corrective action can be taken against presumed data corruption if the check values do not match.

14.1 Introduction

The GPCRC module implements a Cyclic Redundancy Check (CRC) function. It supports both 32-bit and 16-bit polynomials. The supported 32-bit polynomial is 0x04C11DB7(IEEE 802.3), while the 16-bit polynomial can be programmed to any value, depending on the needs of the application. Common 16-bit polynomials are 0x1021 (CCITT-16), 0x3D65 (IEC16-MBus), and 0x8005 (zigbee, 802.15.4, and USB).

14.2 Features

- Programmable 16-bit polynomial, fixed 32-bit polynomial
- Byte-level bit reversal for the CRC input
- Byte-order reorientation for the CRC input
- Word or half-word bit reversal of the CRC result
- Ability to configure and seed an operation in a single register write
- Single-cycle CRC computation for 32-, 16-, or 8-bit blocks
- DMA operation

14.3 Functional Description

An overview of the GPCRC module is shown in Figure 14.1 GPCRC Overview on page 400.

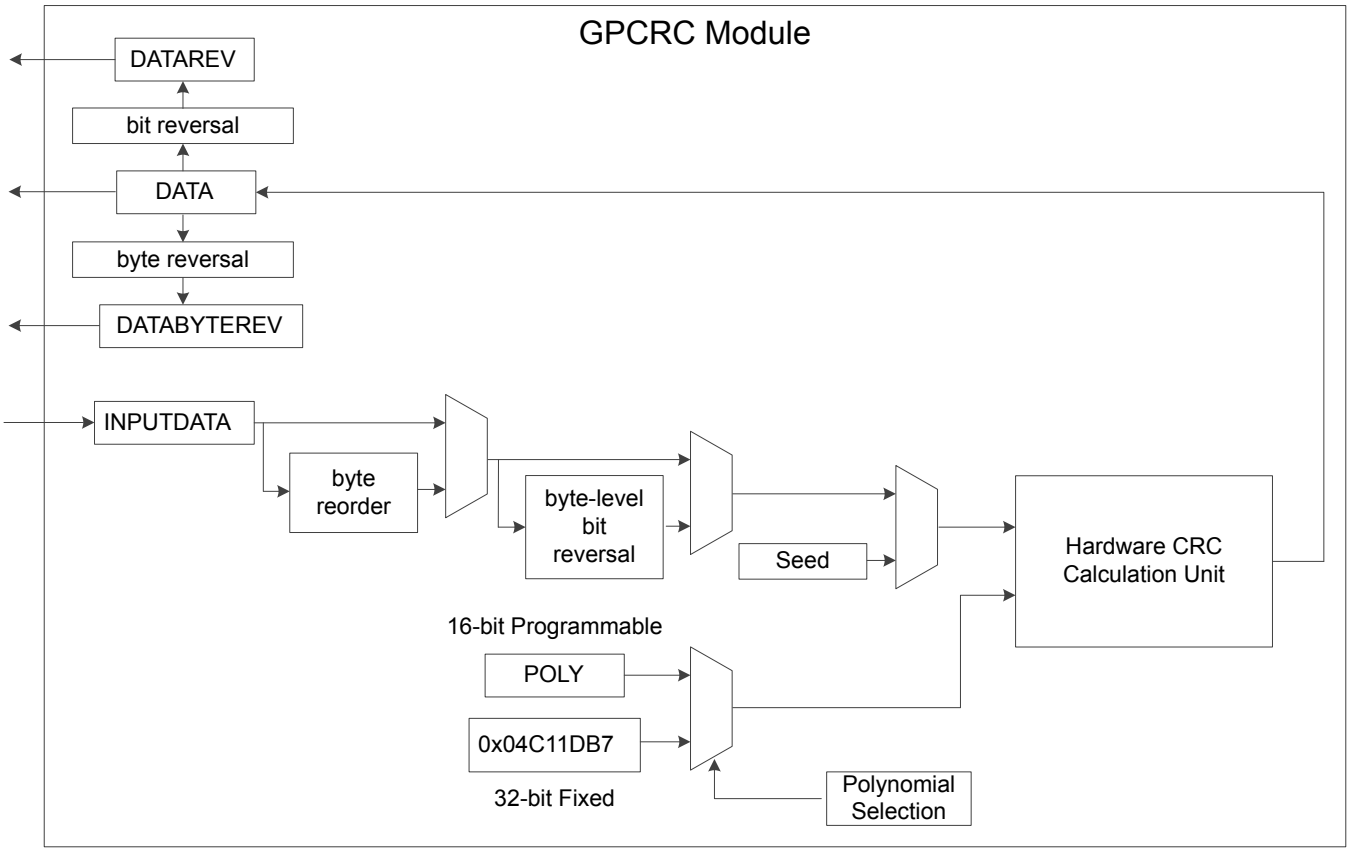


Figure 14.1. GPCRC Overview

14.3.1 Polynomial Specification

POLYSEL in GPCRC_CTRL selects between 32-bit and 16-bit polynomial functions. When a 32-bit polynomial is selected, the fixed IEEE 802.3 polynomial(0x04C11DB7) is used. When a 16-bit polynomial is selected, any valid polynomial can be defined by the user in GPCRC_POLY.

A valid 16-bit CRC polynomial must have an x^16 term and an x^0 term. Theoretically, a 16-bit polynomial has 17 terms total. The convention used is to omit the x^16 term. The polynomial should be written in **reversed** (little endian) bit order. The most significant bit corresponds to the lowest order term. Thus, the most significant bit in CRC_POLY represents the x^0 term, and the least significant bit in CRC_POLY represents the x^15 term. The highest significant bit of CRC_POLY should always set to 1.The polynomial representation for the CRC-16-CCIT polynomial x^16 + x^12 + x^5 + 1, or 0x8408 in reversed order, is shown in Figure 14.2 Polynomial Representation on page 401.

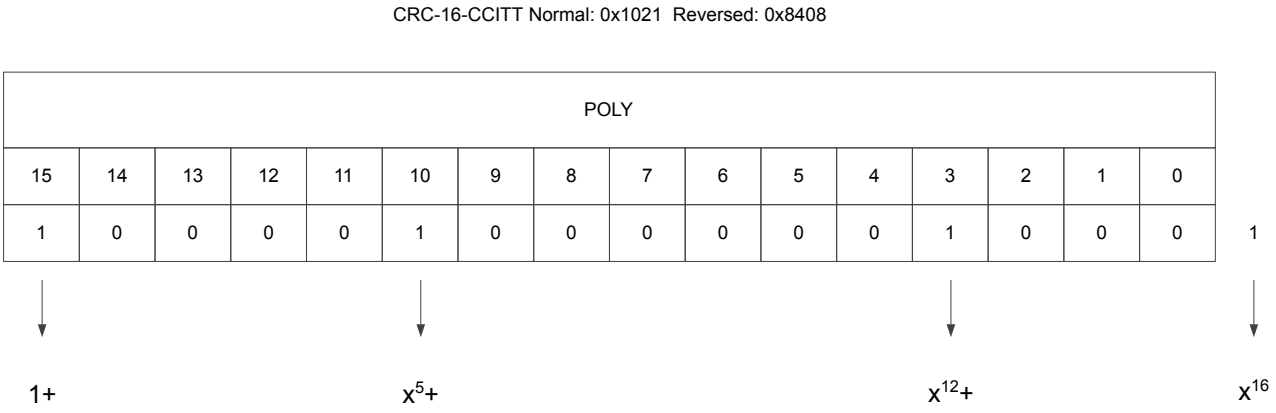


Figure 14.2. Polynomial Representation

14.3.2 Input and Output Specification

The CRC input data can be written to the GPCRC_INPUTDATA, GPCRC_INPUTDATAHWORD or GPCRC_INPUTDATABYTE register via the APB bus based on different data size. If BYTEMODE in GPCRC_CTRL is set, only the least significant byte of the data word will be used for the CRC calculation no matter which input register is written. There are also three output registers for different ordering. Reading from GPCRC_DATA will get the result based on the polynomial in reversed order, while reading from GPCRC_DATAREV will get the result based on the polynomial in normal order. The CRC calculation completes in one clock cycle. Reads from the GPCRC_DATA, GPCRC_DATAREV or GPCRC_DATABYTEREV registers and writes to the GPCRC_CMD register are halted while the calculation is in progress.

.

14.3.3 Initialization

The CRC can be pre-loaded or re-initialized by first writing a 32-bit programmable init value to INIT in GPCRC_INIT and then setting INIT in GPCRC_CMD. It can also be re-initialized automatically when read from DATA, DATAREV or DATABYTEREV provided that AUTOINIT in GPCRC_CTRL is set, the CRC would be re-initialized with the stored init value.

14.3.4 DMA Usage

A DMA channel may be used to transfer data into the CRC engine. All bytes and half-word writes must be word-aligned. The recommended DMA usage model is to use the DMA to transfer all available words of data and use software writes to capture any remaining bytes.

14.3.5 Byte-Level Bit Reversal and Byte Reordering

The byte-level bit reversal and byte reordering operations occur before the data is used in the CRC calculation. Byte reordering can occur on words or half words. The hardware ignores the BYTEREVERSE field with any byte writes or operations with byte mode enabled (BYTEMODE = 1), but the bit reversal settings (BITREVERSE) are still applied to the byte. 32-bit little endian MSB-first data can be treated like 32-bit little endian LSB-first data, as shown in Figure 14.3 Data Ordering Example - 32-bit MSB -first to LSB-first on page 402. In this example, 32-bit data is written to GPCRC_INPUTDATA, BYTEREVERSE is set for byte ordering, and BITREVERSE is set for byte-level bit reversal.

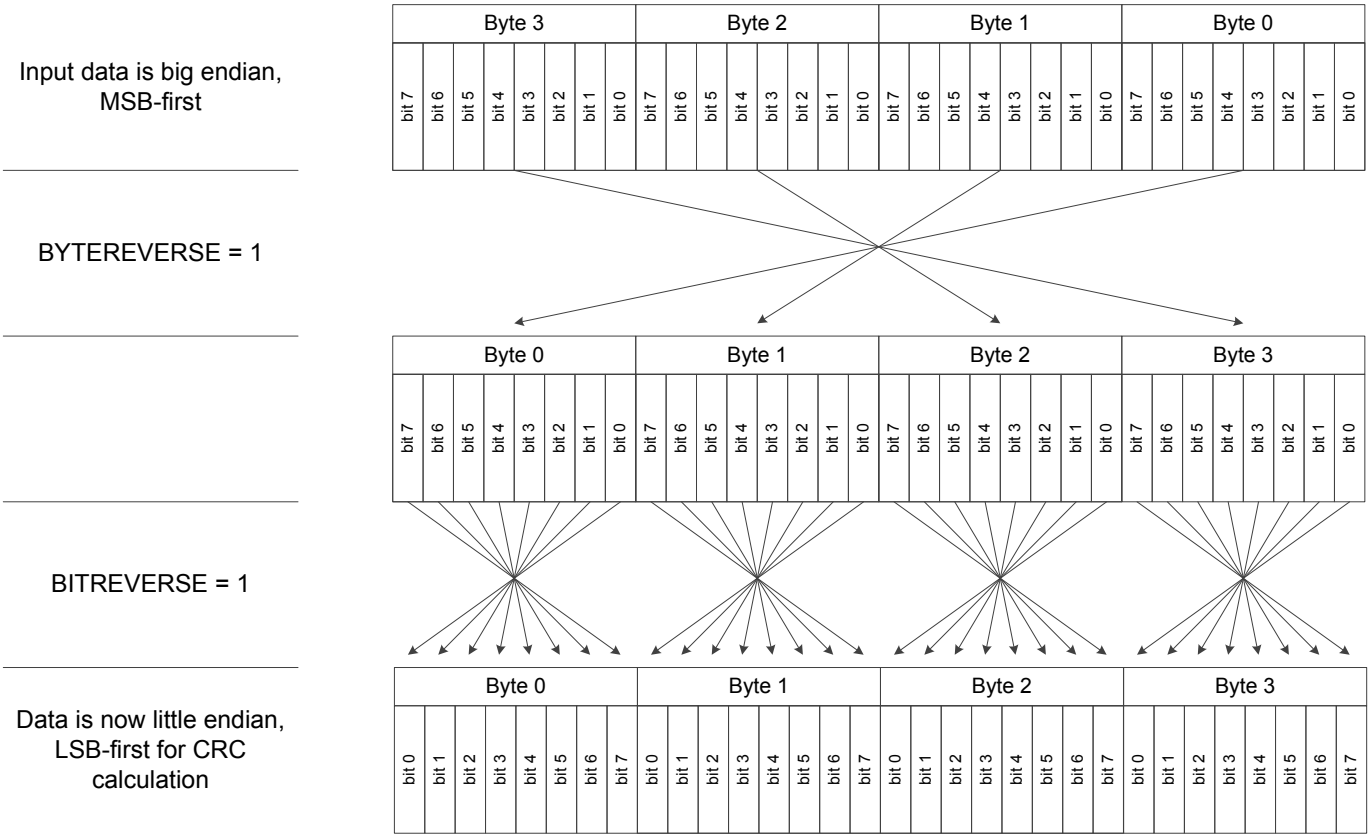


Figure 14.3. Data Ordering Example - 32-bit MSB -first to LSB-first

When handling 16-bit data, the byte reordering function only swap the two lowest bytes and clear the two highest bytes, as shown in Figure 14.4 Data Ordering Example - 16-bit MSB -first to LSB-first on page 403. In this example, 16-bit data is written to GPCRC_INPUTDATAHWORD, BYTEREVERSE is set for byte ordering, and BITREVERSE is set for byte-level bit reversal.

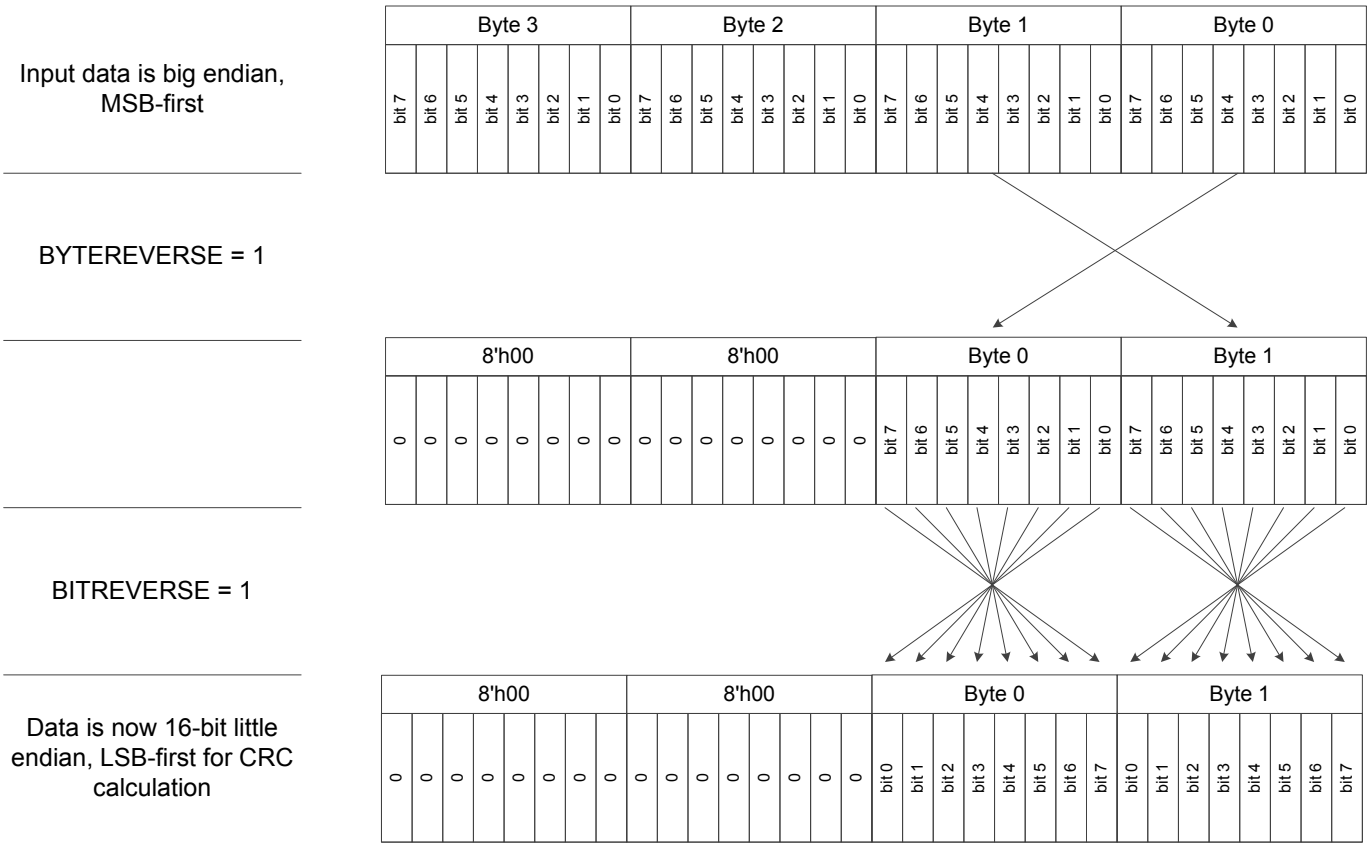


Figure 14.4. Data Ordering Example - 16-bit MSB -first to LSB-first

Assuming a word input byte order of B3 B2 B1 B0, the values used in the CRC calculation for the various settings of the byte-level bit reversal and byte reordering are shown in Table 14.1 Byte-Level Bit Reversal and Byte Reordering Results (B3 B2 B1 B0 Input Order) on page 403.

Table 14.1. Byte-Level Bit Reversal and Byte Reordering Results (B3 B2 B1 B0 Input Order)

Input Width(bits)	BYTEREVERSE Setting	BITREVERSE Setting	Input to CRC Calculation
32	0	0	B3 B2 B1 B0
32	1	1	'B0 'B1 'B2 'B3
32	1	0	B0 B1 B2 B3
32	0	1	'B3 'B2 'B1 'B0
16	0	0	XX XX B1 B0
16	1	1	XX XX 'B0 'B1
16	1	0	XX XX B0 B1
16	0	1	XX XX 'B1 'B0
8	-	0	XX XX XX XX B0
8	-	1	XX XX XX XX 'B0

Input Width(bits)	BYTEREVERSE Setting	BITREVERSE Setting	Input to CRC Calculation

Notes:

1. X indicates a "don't care".
2. Bn is the byte field within the word.
3. 'Bn is the bit-reversed byte field within the word.

14.4 GPCRC Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	GPCRC_IPVERSION	R	IP Version ID
0x004	GPCRC_EN	RW	CRC Enable
0x008	GPCRC_CTRL	RW	Control Register
0x00C	GPCRC_CMD	W	Command Register
0x010	GPCRC_INIT	RWH	CRC Init Value
0x014	GPCRC_POLY	RW	CRC Polynomial Value
0x018	GPCRC_INPUTDATA	W	Input 32-bit Data Register
0x01C	GPCRC_INPUTDATAHWORD	W	Input 16-bit Data Register
0x020	GPCRC_INPUTDATABYTE	W	Input 8-bit Data Register
0x024	GPCRC_DATA	R(r)H	CRC Data Register
0x028	GPCRC_DATAREV	R(r)H	CRC Data Reverse Register
0x02C	GPCRC_DATABYTEREV	R(r)H	CRC Data Byte Reverse Register
0x1000	GPCRC_IPVERSION_SET	R	IP Version ID
0x1004	GPCRC_EN_SET	RW	CRC Enable
0x1008	GPCRC_CTRL_SET	RW	Control Register
0x100C	GPCRC_CMD_SET	W	Command Register
0x1010	GPCRC_INIT_SET	RWH	CRC Init Value
0x1014	GPCRC_POLY_SET	RW	CRC Polynomial Value
0x1018	GPCRC_INPUTDATA_SET	W	Input 32-bit Data Register
0x101C	GPCRC_INPUTDATAH-WORD_SET	W	Input 16-bit Data Register
0x1020	GPCRC_INPUTDATA-BYTE_SET	W	Input 8-bit Data Register
0x1024	GPCRC_DATA_SET	R(r)H	CRC Data Register
0x1028	GPCRC_DATAREV_SET	R(r)H	CRC Data Reverse Register
0x102C	GPCRC_DATABYTEREV_SET	R(r)H	CRC Data Byte Reverse Register
0x2000	GPCRC_IPVERSION_CLR	R	IP Version ID
0x2004	GPCRC_EN_CLR	RW	CRC Enable
0x2008	GPCRC_CTRL_CLR	RW	Control Register
0x200C	GPCRC_CMD_CLR	W	Command Register
0x2010	GPCRC_INIT_CLR	RWH	CRC Init Value
0x2014	GPCRC_POLY_CLR	RW	CRC Polynomial Value
0x2018	GPCRC_INPUTDATA_CLR	W	Input 32-bit Data Register
0x201C	GPCRC_INPUTDATAH-WORD_CLR	W	Input 16-bit Data Register
0x2020	GPCRC_INPUTDATA-BYTE_CLR	W	Input 8-bit Data Register

Offset	Name	Type	Description
0x2024	GPCRC_DATA_CLR	R(r)H	CRC Data Register
0x2028	GPCRC_DATAREV_CLR	R(r)H	CRC Data Reverse Register
0x202C	GPCRC_DATABYTEREV_CLR	R(r)H	CRC Data Byte Reverse Register
0x3000	GPCRC_IPVERSION_TGL	R	IP Version ID
0x3004	GPCRC_EN_TGL	RW	CRC Enable
0x3008	GPCRC_CTRL_TGL	RW	Control Register
0x300C	GPCRC_CMD_TGL	W	Command Register
0x3010	GPCRC_INIT_TGL	RWH	CRC Init Value
0x3014	GPCRC_POLY_TGL	RW	CRC Polynomial Value
0x3018	GPCRC_INPUTDATA_TGL	W	Input 32-bit Data Register
0x301C	GPCRC_INPUTDATAH-WORD_TGL	W	Input 16-bit Data Register
0x3020	GPCRC_INPUTDATA-BYTE_TGL	W	Input 8-bit Data Register
0x3024	GPCRC_DATA_TGL	R(r)H	CRC Data Register
0x3028	GPCRC_DATAREV_TGL	R(r)H	CRC Data Reverse Register
0x302C	GPCRC_DATABYTEREV_TGL	R(r)H	CRC Data Byte Reverse Register

14.5 GPCRC Register Description

14.5.1 GPCRC_IPVERSION - IP Version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP Version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

14.5.2 GPCRC_EN - CRC Enable

Offset	Bit Position																														
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**CRC Enable**

The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.

Value	Mode	Description
0	DISABLE	Disable CRC function. Reordering functions are still available. Only BITREVERSE and BYTEREVERSE bits are configurable in this mode.
1	ENABLE	Writes to INPUTDATA registers will result in CRC operations.

14.5.3 GPCRC_CTRL - Control Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x0			0x0	0x0	0x0				0x0				
Access																			RW			RW	RW	RW				RW				
Name																			AUTOINIT			BYTEREVERSE	BITREVERSE	BYTEMODE				POLYSEL				

Bit	Name	Reset	Access	Description
31:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	AUTOINIT	0x0	RW	**Auto Init Enable**
	Enables auto init by re-seeding the CRC result based on the value in INIT after reading of DATA, DATAREV or DATABYTEREV.			
12:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	BYTEREVERSE	0x0	RW	**Byte Reverse Mode**
	Allows byte level reverse of bytes B3, B2, B1, B0 within the 32-bit data word			

Value	Mode	Description
0	NORMAL	No reverse: B3, B2, B1, B0
1	REVERSED	Reverse byte order. For 32-bit: B0, B1, B2, B3; For 16-bit: 0, 0, B0, B1

Bit	Name	Reset	Access	Description
9	BITREVERSE	0x0	RW	**Byte-level Bit Reverse Enable**
	Reverses bits within each byte of the 32-bit data word			

Value	Mode	Description
0	NORMAL	No reverse
1	REVERSED	Reverse bit order in each byte

Bit	Name	Reset	Access	Description
8	BYTEMODE	0x0	RW	**Byte Mode Enable**
	Treats all writes as bytes. Only the least significant byte of the data-word will be used for CRC calculation for all writes			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	POLYSEL	0x0	RW	**Polynomial Select**
	Selects 16-bit CRC programmable polynomial or 32-bit CRC fixed polynomial			

Value	Mode	Description
0	CRC32	CRC-32 (0x04C11DB7) polynomial selected
1	CRC16	16-bit CRC programmable polynomial selected

Bit	Name	Reset	Access	Description
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

14.5.4 GPCRC_CMD - Command Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																W
Name																																INIT

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	INIT	0x0	W	**Initialization Enable**
	Writing 1 to this bit initialize the CRC by writing the INIT value in CRC_INIT to CRC_DATA.			

14.5.5 GPCRC_INIT - CRC Init Value

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	INIT																															

Bit	Name	Reset	Access	Description
31:0	INIT	0x0	RW	**CRC Initialization Value**
	This value is loaded into CRC_DATA upon issuing the INIT command in CRC_CMD			

14.5.6 GPCRC_POLY - CRC Polynomial Value

Offset	Bit Position																															
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	RW															
Name																	POLY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	POLY	0x0	RW	**CRC Polynomial Value**
	This value defines 16-bit POLY, which is used as the polynomial during the 16-bit CRC calculation. The polynomial is defined in reversed representation, meaning that the lowest degree term is in the highest bit position of POLY. Additionally, the highest degree term in the polynomial is implicit. Further examples of the CRC configuration can be found in the documentation.			

14.5.7 GPCRC_INPUTDATA - Input 32-bit Data Register

Offset	Bit Position																															
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0																
Access																W																
Name																INPUTDATA																

Bit	Name	Reset	Access	Description
31:0	INPUTDATA	0x0	W	**Input Data for 32-bit**
	CRC Input 32-bit Data can be written to this register. Each time this register is written, the CRC value is updated.			

14.5.8 GPCRC_INPUTDATAHWORD - Input 16-bit Data Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	W															
Name																	INPUTDATAHWORD															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	INPUTDATAHWORD	0x0	W	**Input Data for 16-bit**
	CRC Input 16-bit Data can be written to this register. Each time this register is written, the CRC value is updated.			

14.5.9 GPCRC_INPUTDATABYTE - Input 8-bit Data Register

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									INPUTDATABYTE							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	INPUTDATABYTE	0x0	W	**Input Data for 8-bit**
	CRC Input 8-bit Data can be written to this register. Each time this register is written, the CRC value is updated.			

14.5.10 GPCRC_DATA - CRC Data Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R(r)																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x0	R(r)	**CRC Data Register**
	CRC Data Register, read only. The CRC data register may still be indirectly written from software, by writing the INIT register and then issue an INITIALIZE command.			

14.5.11 GPCRC_DATAREV - CRC Data Reverse Register

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R(r)																															
Name	DATAREV																															

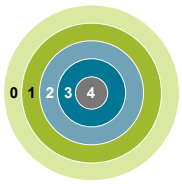
Bit	Name	Reset	Access	Description
31:0	DATAREV	0x0	R(r)	**Data Reverse Value**
	Bit reversed version of CRC Data register. When a 32-bit CRC polynomial is selected, the reversal occurs on the entire 32-bit word. When a 16-bit CRC polynomial is selected, the bits [15:0] are reversed.			

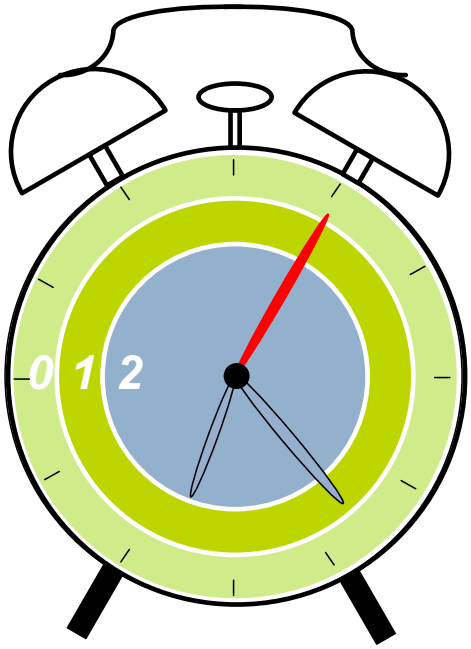
14.5.12 GPCRC_DATABYTEREV - CRC Data Byte Reverse Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R(r)																															
Name	DATABYTEREV																															

Bit	Name	Reset	Access	Description
31:0	DATABYTEREV	0x0	R(r)	**Data Byte Reverse Value**
	Byte reversed version of CRC Data register. When a 32-bit CRC polynomial is selected, the bytes are swizzled to {B0, B1, B2, B3}. When a 16-bit CRC polynomial is selected, the bytes are swizzled to {0, 0, B0, B1}.			

15. RTCC - Real Time Clock with Capture





Quick Facts

What?

The Real Time Clock with Capture (RTCC)) is a 32-bit Real Time Clock ensuring timekeeping in low energy modes.

Why?

Timekeeping over long time periods while using as little power as possible is required in many low power applications.

How?

A low frequency oscillator is used as clock signal and the RTCC has three different Capture/Compare channels which can trigger wake-up, generate PRS signalling, or capture system events. 32-bit resolution and selectable prescaling allow the system to stay in low energy modes for long periods of time and still maintain reliable timekeeping.

15.1 Introduction

The Real Time Clock with Capture (RTCC), with three capture/compare channels, is a 32-bit counter kept running down to energy mode EM3. It can be used as an EM2/3 wakeup source as well as a timekeeping counter during low energy mode. Time keeping over long time periods while using as little power as possible is required in many low-power applications. The 32-bit counter is in combination with a 15-bit pre-counter to allow flexible pre-scaling of the main counter.

Three individually configurable Capture/Compare channels can be used to trigger interrupts, generate PRS signals, capture system events, and to wake the device up from EM2, or EM3 when using the ULFRCO as a clock source.

15.2 Features

A low frequency oscillator is used as clock signal and the RTCC has three different Capture/Compare channels which can trigger wake-up, generate PRS signalling, or capture system events. 32-bit resolution and selectable pre-scaling allows the system to stay in low energy modes for long periods of time and still maintain reliable timekeeping.

- 32-bit Real Time Counter
- 15-bit pre-counter, for flexible frequency scaling or for use as an independent counter
- EM2/EM3 operation and wakeup (EM3 when using ULFRCO as clock source)
- Can survive system reset, only POR and EM4 wakeup will reset RTCC
- Three Capture/Compare channels
  - Capture of PRS events from other parts of the system, value stored in ICVALUE
  - Compare match or input capture can trigger interrupts
  - Compare channel 1, RTCC_CC1_OCVALUE can be used as a top value for the main counter
  - Compare channel 0, RTCC_CC0_OCVALUE can be used as a top value for the pre-counter
  - Compare match events are available to other peripherals through the Peripheral Reflex System (PRS)

15.3 Functional Description

An overview of the RTCC is shown in Figure 15.1 RTCC Overview on page 415 .

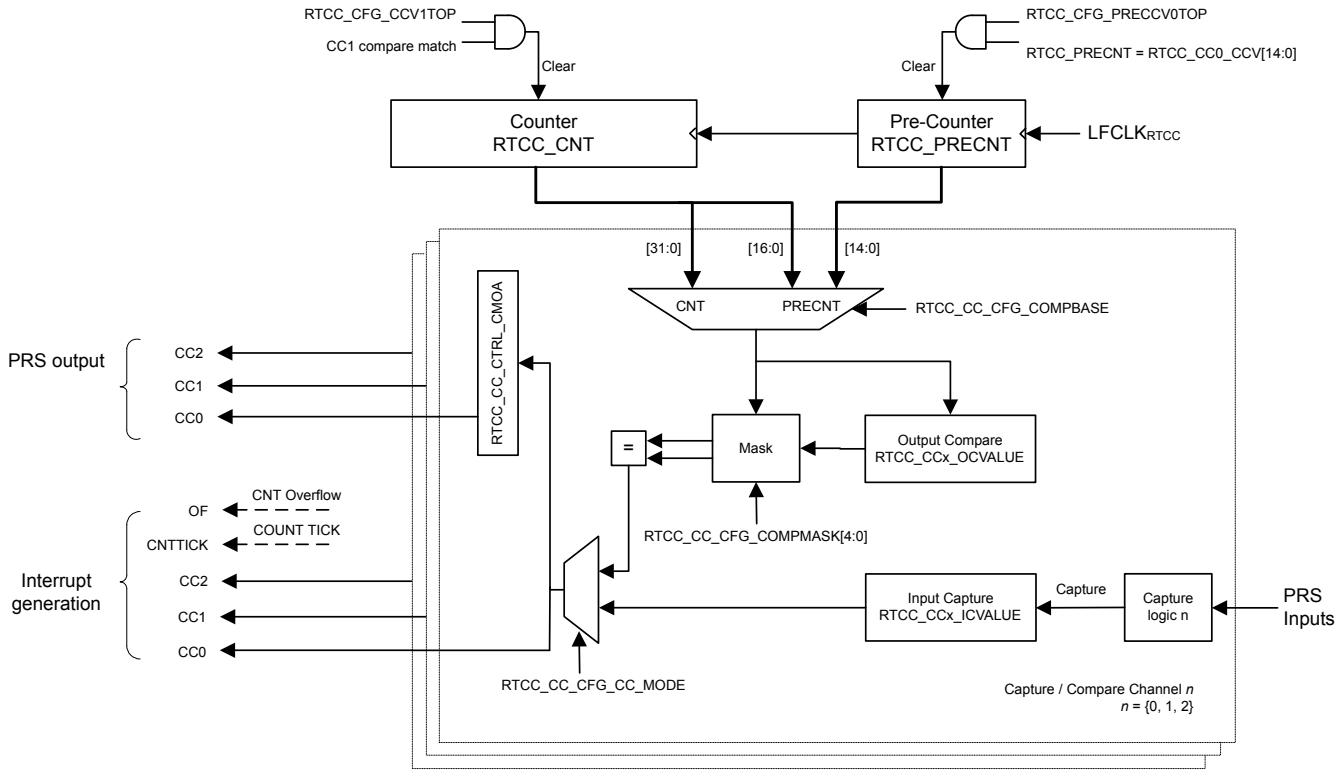


Figure 15.1. RTCC Overview

15.3.1 RTCC Counter

The RTCC consists of two counters; the 32-bit main counter, RTCC_CNT, and a 15-bit pre-counter, RTCC_PRECNT. The pre-counter can be used as an independent counter, or to generate a specific frequency for the main counter. In both configurations, the pre-counter can be used to generate compare match events or be captured in the Capture/Compare channels as a result of an external PRS event. Refer to Capture/Compare Channels for details on how to configure the Capture/Compare channels for use with the pre-counter.

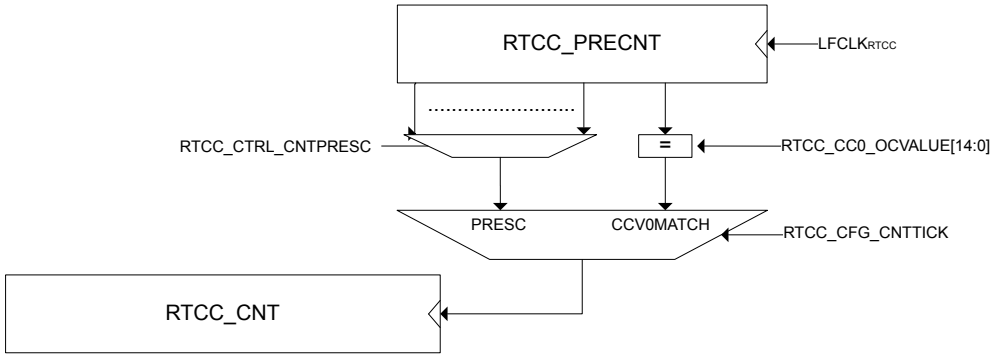


Figure 15.2. RTCC counter Block Diagram

The RTCC peripheral clock is requested by setting the EN bit in RTCC_EN. Then RTCC can be enabled by setting the command register START in RTCC_CMD. When the RTCC is enabled, the pre-counter (RTCC_PRECNT) increments upon each positive clock edge of low frequency clock. If CNTTICK in RTCC_CFG is set to PRESC, the pre-counter will continue to count up, wrapping around to zero when it overflows. If CNTTICK in RTCC_CFG is set to CCV0MATCH, the pre-counter will wrap around when it hits the value configured in RTCC_CC0_OCVALUE.

The main counter is available in RTCC_CNT and increments upon each tick given from the pre-counter. Refer to Normal Mode for a description on how to configure the frequency of these ticks. The main counter can receive a tick based on different dividers from the pre-counter, allowing the ticks to be power of 2 divisions of the LF clock. For more accurate configuration of the tick frequency, RTCC_CC0_OCVALUE[14:0] can be used as a top value for
RTCC_PRECNT. When reaching the top value, the main counter receives a tick, and the pre-counter wraps around. Table below shows RTCC Resolution vs Overflow, FLFCLK = 32768 Hz, which summarizes the resolutions available when using a 32768 Hz oscillator as source for LF clock of RTCC.

Table 15.1. RTCC Resolution vs Overflow, F_{LFCLK} = 32768 Hz

RTCC_CTRL_CNTTICK	RTCC_CTRL_CNTPRESC	Main counter period, T_{CNT}	Overflow
CCV0MATCH	Don't care	(RTCC_CC0_OCVALUE + 1)/F_{LFCLK} s	2^{32}*T_{CNT} seconds

RTCC_CTRL_CNTTICK	RTCC_CTRL_CNTPRESC	Main counter period, T_{CNT}	Overflow
PRESC	DIV1	30.5 µs	36.4 hours
	DIV2	61 µs	72.8 hours
	DIV4	122 µs	145.6 hours
	DIV8	244 µs	12 days
	DIV16	488 µs	24 days
	DIV32	977 µs	48 days
	DIV64	1.95 ms	97 days
	DIV128	3.91 ms	194 days
	DIV256	7.81 ms	388 days
	DIV512	15.6 ms	776 days
	DIV1024	31.25 ms	4.2 years
	DIV2048	62.5 ms	8.5 years
	DIV4096	0.125 s	17 years
	DIV8192	0.25 s	34 years
	DIV16384	0.5 s	68 years
	DIV32768	1 s	136 years

By default, the counter will keep counting until it reaches the top value, 0xFFFFFFFF, before it wraps around and continues counting from zero. By setting CCV1TOP in RTCC_CFG, a Capture/Compare channel 1 compare match will result in the main counter wrapping to 0. The timer will then wrap around on a channel 1 compare match (RTCC_CNT = RTCC_CC1_OCVALUE). If using the CCV1TOP setting, make sure to set this bit prior to or at the same time the RTCC is enabled. Setting CCV1TOP after enabling the RTCC may cause unintended operation (e.g. if RTCC_CNT > RTCC_CC1_OCVALUE, RTCC_CNT will wrap when reaching 0xFFFFFFFF rather than RTCC_CC1_OCVALUE).

The counters of the RTCC, RTCC_CNT and RTCC_PRECNT, can at any time be written by software, as long as the registers are not locked using RTCC_LOCKKEY. All RTCC registers use the new immediate synchronization scheme.

Note: Writing to the RTCC_PRECNT register may alter the frequency of the ticks for the RTCC_CNT register.

15.3.2 Capture/Compare Channels

Three capture/compare channels are available in the RTCC. Each channel can be configured as input capture or output compare, by setting the corresponding MODE in the RTCC_CCx_CTRL register.

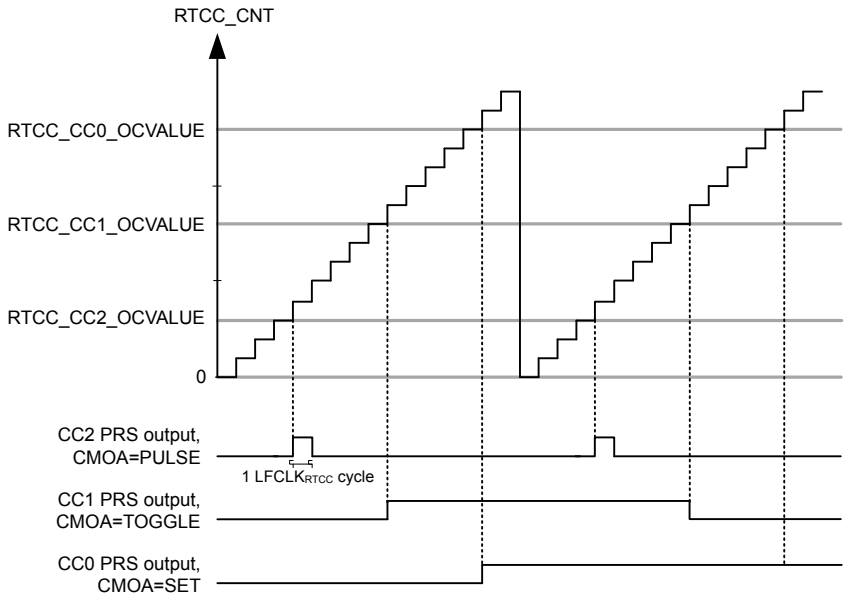


Figure 15.3. RTCC Compare Match and PRS Output Illustration

In input capture mode the RTCC_CNT register is captured into the RTCC_CCx_ICVALUE register when an edge is detected on the selected PRS input channel. The active capture edge is configured in the ICEDGE control bits.

In output compare mode the compare values are set by writing to the RTCC compare channel registers RTCC_CCx_OCVALUE. These values will be compared to the main counter, RTCC_CNT or a mixture of the main counter and the pre-counter, as illustrated in Figure 15.4 RTCC Compare Base Illustration on page 419. Compare base for the capture compare channels is set by configuring COMPBASE in RTCC_CCx_CTRL.

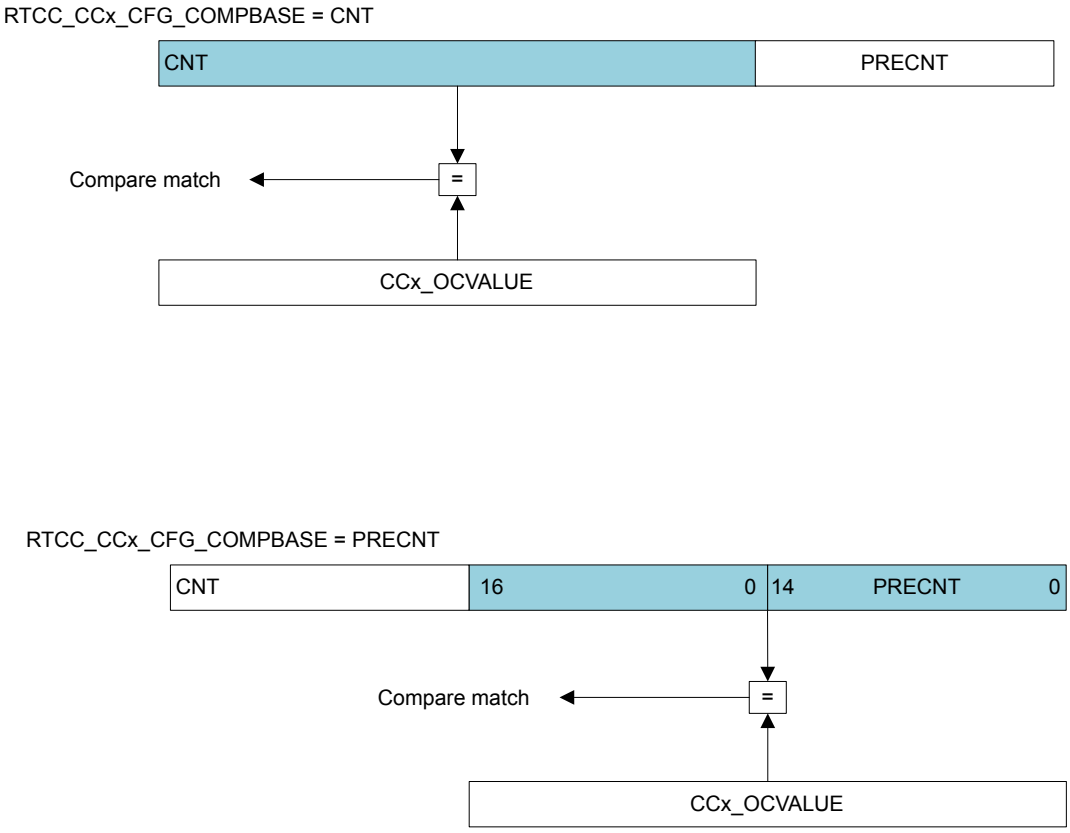


Figure 15.4. RTCC Compare Base Illustration

Table RTCC Capture/Compare subjects summarizes which registers being subject to comparison for different configurations of RTCC_CTRL_CNTMODE and RTCC_CCx_CTRL_COMPBASE.

Table 15.2. RTCC Capture/Compare subjects

RTCC_CTRL_CNTMODE	NORMAL
RTCC_CCx_CTRL_COMPBASE = CNT	RTCC_CNT vs. RTCC_CCx_OCVALUE
RTCC_CCx_CTRL_COMPBASE = PRECNT	{RTCC_CNT[16:0],RTCC_PRECNT[14:0]} vs. RTCC_CCx_OCVALUE

15.3.3 Interrupts and PRS Output

The RTCC has interrupts for each of its 3 Capture/Compare channels (CC0, CC1, and CC2), as well as a counter tick interrupt (CNTTICK) and an overflow interrupt (OF). The counter tick interrupt is set each time the main counter receives a tick, while the overflow interrupt occurs when the main counter overflows.

Each Capture/Compare channel has a PRS output with configurable actions upon compare match. The output action is determined by the CMOA field in register RTCC_CCx_CTRL. See 13.3.3 Producers for more details on how to connect PRS channels to these outputs.

15.3.4 Register Lock

To prevent accidental writes to the RTCC registers, the RTCC_LOCK register can be written to any other value than the unlock value. To unlock the register, write the unlock value to RTCC_LOCKKEY. Registers affected by this lock are:

- RTCC_CFG
- RTCC_EN
- RTCC_CMD
- RTCC_PRECNT
- RTCC_CNT
- RTCC_CCx_CTRL
- RTCC_CCx_OCVALUE
- RTCC_CCx_ICVALUE

15.3.5 Programmer's Model

The registers of RTCC can be divided into a few groups as below,

CFG: config registers
EN: enable register to make the peripheral clock available to RTCC
CTRL: control or other registers can be programmed during run
CMD: command registers to start/stop RTCC running
STATUS: read only status registers

Generally speaking, in order to use and program RTCC properly, it should follow the sequence below,

Set CFG->Set EN->Set CTRL->START CMD->adjust CTRL->STOP CMD

All the registers have been separated into different synchronization types. The CFG register is WSTATIC, which means only when EN=0, it will allow the programming of CFG, otherwise there will be a bus fault for the CFG register write. Here is an example of programming CFG prior to setting EN to 1.

```
RTCC->CC[0].OCVALUE = 2;
    RTCC->CC[0].CTRL = RTCC_CC_CTRL_CC_MODE_OUTPUTCOMPARE;
    RTCC->CC[0].OCVALUE = 5;
    RTCC->EN = RTCC_EN_EN;
```

All the other registers with low frequency synchronization types need to be programmed after setting EN to 1. Counter will only start to count once START command is issued. For LFRWSYNC registers, user needs to keep polling sync busy, e.g. START, before programming the same register once again.

```
// Bang on start till it's running
      do {
        RTCC->CMD = RTCC_CMD_START;
        while(RTCC->SYNCBUSY & _RTCC_SYNCBUSY_MASK);
      } while ( (RTCC->STATUS & _RTCC_STATUS_RUNNING_MASK) != RTCC_STATUS_RUNNING );
```

For QUICKLFWSYNC registers, when writing to it, the write will stall the bus until the write action is completed, so there is no sync busy bit for those registers, RTCC_CCn_CTRL is an example for that.

15.3.6 Debug Features and Description

By default, the RTCC is halted when code execution is halted from the debugger. By setting the DEBUGRUN bit in the RTCC_CTRL register, the RTCC will continue to run even when the debugger has halted the system.

15.4 RTCC Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	RTCC_IPVERSION	R	IP VERSION
0x004	RTCC_EN	RW ENABLE	Module Enable Register
0x008	RTCC_CFG	RW CONFIG	Configuration Register
0x00C	RTCC_CMD	W LFSYNC	Command Register
0x010	RTCC_STATUS	RH	Status register
0x014	RTCC_IF	RWH INTFLAG	RTCC Interrupt Flags
0x018	RTCC_IEN	RW	Interrupt Enable Register
0x01C	RTCC_PRECNT	RWH LFSYNC	Pre-Counter Value Register
0x020	RTCC_CNT	RWH LFSYNC	Counter Value Register
0x024	RTCC_COMBCNT	RH	Combined Pre-Counter and Counter Valu...
0x028	RTCC_SYNCBUSY	RH	Synchronization Busy Register
0x02C	RTCC_LOCK	W	Configuration Lock Register
0x030	RTCC_CCx_CTRL	RW	CC Channel Control Register
0x034	RTCC_CCx_OCVALUE	RW	Output Compare Value Register
0x038	RTCC_CCx_ICVALUE	RH	Input Capture Value Register
0x1000	RTCC_IPVERSION_SET	R	IP VERSION
0x1004	RTCC_EN_SET	RW ENABLE	Module Enable Register
0x1008	RTCC_CFG_SET	RW CONFIG	Configuration Register
0x100C	RTCC_CMD_SET	W LFSYNC	Command Register
0x1010	RTCC_STATUS_SET	RH	Status register
0x1014	RTCC_IF_SET	RWH INTFLAG	RTCC Interrupt Flags
0x1018	RTCC_IEN_SET	RW	Interrupt Enable Register
0x101C	RTCC_PRECNT_SET	RWH LFSYNC	Pre-Counter Value Register
0x1020	RTCC_CNT_SET	RWH LFSYNC	Counter Value Register
0x1024	RTCC_COMBCNT_SET	RH	Combined Pre-Counter and Counter Valu...
0x1028	RTCC_SYNCBUSY_SET	RH	Synchronization Busy Register
0x102C	RTCC_LOCK_SET	W	Configuration Lock Register
0x1030	RTCC_CCx_CTRL_SET	RW	CC Channel Control Register
0x1034	RTCC_CCx_OCVALUE_SET	RW	Output Compare Value Register
0x1038	RTCC_CCx_ICVALUE_SET	RH	Input Capture Value Register
0x2000	RTCC_IPVERSION_CLR	R	IP VERSION
0x2004	RTCC_EN_CLR	RW ENABLE	Module Enable Register
0x2008	RTCC_CFG_CLR	RW CONFIG	Configuration Register
0x200C	RTCC_CMD_CLR	W LFSYNC	Command Register
0x2010	RTCC_STATUS_CLR	RH	Status register

Offset	Name	Type	Description
0x2014	RTCC_IF_CLR	RWH INTFLAG	RTCC Interrupt Flags
0x2018	RTCC_IEN_CLR	RW	Interrupt Enable Register
0x201C	RTCC_PRECNT_CLR	RWH LFSYNC	Pre-Counter Value Register
0x2020	RTCC_CNT_CLR	RWH LFSYNC	Counter Value Register
0x2024	RTCC_COMBCNT_CLR	RH	Combined Pre-Counter and Counter Valu...
0x2028	RTCC_SYNCBUSY_CLR	RH	Synchronization Busy Register
0x202C	RTCC_LOCK_CLR	W	Configuration Lock Register
0x2030	RTCC_CCx_CTRL_CLR	RW	CC Channel Control Register
0x2034	RTCC_CCx_OCVALUE_CLR	RW	Output Compare Value Register
0x2038	RTCC_CCx_ICVALUE_CLR	RH	Input Capture Value Register
0x3000	RTCC_IPVERSION_TGL	R	IP VERSION
0x3004	RTCC_EN_TGL	RW ENABLE	Module Enable Register
0x3008	RTCC_CFG_TGL	RW CONFIG	Configuration Register
0x300C	RTCC_CMD_TGL	W LFSYNC	Command Register
0x3010	RTCC_STATUS_TGL	RH	Status register
0x3014	RTCC_IF_TGL	RWH INTFLAG	RTCC Interrupt Flags
0x3018	RTCC_IEN_TGL	RW	Interrupt Enable Register
0x301C	RTCC_PRECNT_TGL	RWH LFSYNC	Pre-Counter Value Register
0x3020	RTCC_CNT_TGL	RWH LFSYNC	Counter Value Register
0x3024	RTCC_COMBCNT_TGL	RH	Combined Pre-Counter and Counter Valu...
0x3028	RTCC_SYNCBUSY_TGL	RH	Synchronization Busy Register
0x302C	RTCC_LOCK_TGL	W	Configuration Lock Register
0x3030	RTCC_CCx_CTRL_TGL	RW	CC Channel Control Register
0x3034	RTCC_CCx_OCVALUE_TGL	RW	Output Compare Value Register
0x3038	RTCC_CCx_ICVALUE_TGL	RH	Input Capture Value Register

15.5 RTCC Register Description

15.5.1 RTCC_IPVERSION - IP VERSION

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x1	R	**IP VERSION**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

15.5.2 RTCC_EN - Module Enable Register

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**RTCC Enable**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

15.5.3 RTCC_CFG - Configuration Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0				0x0	0x0	0x0	0x0
Access																									RW				RW	RW	RW	RW
Name																									CNTPRESC				CNTTICK	CNTCCV1TOP	PRECNTCCV0TOP	DEBUGRUN

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:4	CNTPRESC	0x0	RW	**Counter prescaler value.**

Configure counting frequency of the CNT register.

Value	Mode	Description
0	DIV1	CLK_CNT = (RTCC LF CLK)/1
1	DIV2	CLK_CNT = (RTCC LF CLK)/2
2	DIV4	CLK_CNT = (RTCC LF CLK)/4
3	DIV8	CLK_CNT = (RTCC LF CLK)/8
4	DIV16	CLK_CNT = (RTCC LF CLK)/16
5	DIV32	CLK_CNT = (RTCC LF CLK)/32
6	DIV64	CLK_CNT = (RTCC LF CLK)/64
7	DIV128	CLK_CNT = (RTCC LF CLK)/128
8	DIV256	CLK_CNT = (RTCC LF CLK)/256
9	DIV512	CLK_CNT = (RTCC LF CLK)/512
10	DIV1024	CLK_CNT = (RTCC LF CLK)/1024
11	DIV2048	CLK_CNT = (RTCC LF CLK)/2048
12	DIV4096	CLK_CNT = (RTCC LF CLK)/4096
13	DIV8192	CLK_CNT = (RTCC LF CLK)/8192
14	DIV16384	CLK_CNT = (RTCC LF CLK)/16384
15	DIV32768	CLK_CNT = (RTCC LF CLK)/32768

3	CNTTICK	0x0	RW	**Counter prescaler mode.**

Select whether the main counter should tick on RTCC_CC0_OCVALUE[14:0] compare match with the pre-counter or tick on a pre-counter tap selected in CNTPRESC bitfield in the RTCC_CTRL register.

Value	Mode	Description
0	PRESC	CNT register ticks according to configuration in CNTPRESC.

Bit	Name	Reset	Access	Description
	1	CCV0MATCH		CNT register ticks when PRECNT matches RTCC_CC0_OC[14:0]
2	CNTCCV1TOP	0x0	RW	**CCV1 top value enable**
	When set, the counter wraps around on a CC1 event			
1	PRECNTCCV0TOP	0x0	RW	**Pre-counter CCV0 top value enable.**
	When set, the pre-counter wraps around when PRECNT equals RTCC_CC0_OCVALUE[14:0].			
0	DEBUGRUN	0x0	RW	**Debug Mode Run Enable**
	Set this bit to keep the RTCC running during a debug halt.			

Value	Mode	Description
0	X0	RTCC is frozen in debug mode
1	X1	RTCC is running in debug mode

15.5.4 RTCC_CMD - Command Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															W	W
Name																															STOP	START

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	STOP	0x0	W	**Stop RTCC main counter**
	Write a 1 to stop the RTCC			
0	START	0x0	W	**Start RTCC main counter**
	Write a 1 to start the RTCC			

15.5.5 RTCC_STATUS - Status register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															R	R
Name																															RTCCLOCKSTATUS	RUNNING

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	RTCCLOCKSTATUS	0x0	R	**Lock Status**
	Indicates the current status of RTCC Lock			

Value	Mode	Description
0	UNLOCKED	RTCC registers are unlocked
1	LOCKED	RTCC registers are locked

0	RUNNING	0x0	R	**RTCC running status**
	Indicates the current status of RTCC running			

15.5.6 RTCC_IF - RTCC Interrupt Flags

Offset	Bit Position																														
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0		0x0		0x0			0x0	0x0
Access																								RW		RW		RW			RW	RW
Name																								CC2		CC1		CC0			CNTTICK	OF

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	CC2	0x0	RW	**CC Channel n Interrupt Flag**
	This bit indicates that there has been an interrupt event on Compare/Capture channel			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	CC1	0x0	RW	**CC Channel n Interrupt Flag**
	This bit indicates that there has been an interrupt event on Compare/Capture channel			
5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	CC0	0x0	RW	**CC Channel n Interrupt Flag**
	This bit indicates that there has been an interrupt event on Compare/Capture channel			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	CNTTICK	0x0	RW	**Main counter tick**
	Set each time the main counter is updated.			
0	OF	0x0	RW	**Overflow Interrupt Flag**
	Set when a RTCC overflow has occurred.			

15.5.7 RTCC_IEN - Interrupt Enable Register

Offset	Bit Position																														
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0		0x0		0x0			0x0	0x0
Access																								RW		RW		RW			RW	RW
Name																								CC2		CC1		CC0			CNTTICK	OF

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	CC2	0x0	RW	**CC Channel n Interrupt Enable**
	Enable CC channel interrupts			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	CC1	0x0	RW	**CC Channel n Interrupt Enable**
	Enable CC channel interrupts			
5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	CC0	0x0	RW	**CC Channel n Interrupt Enable**
	Enable CC channel interrupts			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	CNTTICK	0x0	RW	**CNTTICK Interrupt Enable**
	Enable cnttick interrupt			
0	OF	0x0	RW	**OF Interrupt Enable**
	Enable overflow interrupt			

15.5.8 RTCC_PRECNT - Pre-Counter Value Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																		0x0														
Access																		RW														
Name																		PRECNT														

Bit	Name	Reset	Access	Description
31:15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:0	PRECNT	0x0	RW	**Pre-Counter Value**
	Gives access to the Pre-counter value of the RTCC.			

15.5.9 RTCC_CNT - Counter Value Register

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0																
Access																RW																
Name																CNT																

Bit	Name	Reset	Access	Description
31:0	CNT	0x0	RW	**Counter Value**
	Gives access to the main counter value of the RTCC.			

15.5.10 RTCC_COMBCNT - Combined Pre-Counter and Counter Valu...

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																	0x0														
Access	R																	R													
Name	CNTLSB																	PRECNT														

Bit	Name	Reset	Access	Description
31:15	CNTLSB	0x0	R	**Counter Value**
	Gives access to the 17 LSBs of the main counter, CNT.			
14:0	PRECNT	0x0	R	**Pre-Counter Value**
	Gives access to the pre-counter, PRECNT.			

15.5.11 RTCC_SYNCBUSY - Synchronization Busy Register

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0	0x0	0x0
Access																													R	R	R	R
Name																													CNT	PRECNT	STOP	START

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	CNT	0x0	R	**Sync busy for CNT**
	Last writing of CNT is synchronizing to LF clock			
2	PRECNT	0x0	R	**Sync busy for PRECNT**
	Last writing of PRECNT is synchronizing to LF clock			
1	STOP	0x0	R	**Sync busy for STOP**
	Last writing of STOP is synchronizing to LF clock			
0	START	0x0	R	**Sync busy for START**
	Last writing of START is synchronizing to LF clock			

15.5.12 RTCC_LOCK - Configuration Lock Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	W															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x0	W	**Configuration Lock Key**

Write any other value than the unlock code to lock RTCC_CFG, RTCC_EN, RTCC_CMD, RTCC_PRECNT, RTCC_CNT and RTCC_CCx_XXX registers from editing. Write the unlock code to unlock.

Value	Mode	Description
44776	UNLOCK	Write to unlock RTCC lockable registers

15.5.13 RTCC_CCx_CTRL - CC Channel Control Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																										0x0		0x0	0x0		0x0	
Access																										RW		RW	RW		RW	
Name																										ICEDGE		COMPBASE	CMOA		MODE	

Bit	Name	Reset	Access	Description
31:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:5	ICEDGE	0x0	RW	**Input Capture Edge Select**
	These bits control which edges the PRS edge detector triggers on.			

Value	Mode	Description
0	RISING	Rising edges detected
1	FALLING	Falling edges detected
2	BOTH	Both edges detected
3	NONE	No edge detection, signal is left as it is

Bit	Name	Reset	Access	Description
4	COMPBASE	0x0	RW	**Capture compare channel comparison base.**
	Configure comparison base for compare channel.			

Value	Mode	Description
0	CNT	RTCC_CCx_ICVALUE/OCVALUE is compared with CNT register.
1	PRECNT	Least significant bits of RTCC_CCx_ICVALUE/OCVALUE are compared with COMBCNT.

Bit	Name	Reset	Access	Description
3:2	CMOA	0x0	RW	**Compare Match Output Action**
	Select output action on compare match.			

Value	Mode	Description
0	PULSE	A single clock cycle pulse is generated on output
1	TOGGLE	Toggle output on compare match
2	CLEAR	Clear output on compare match
3	SET	Set output on compare match

Bit	Name	Reset	Access	Description
1:0	MODE	0x0	RW	**CC Channel Mode**
	These bits select the mode for Compare/Capture channel.			

Value	Mode	Description
0	OFF	Compare/Capture channel turned off

Bit	Name	Reset	Access	Description
	1	INPUTCAPTURE		Input capture
	2	OUTPUTCOMPARE		Output compare

15.5.14 RTCC_CCx_OCVALUE - Output Compare Value Register

Offset	Bit Position																														
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	OC																															

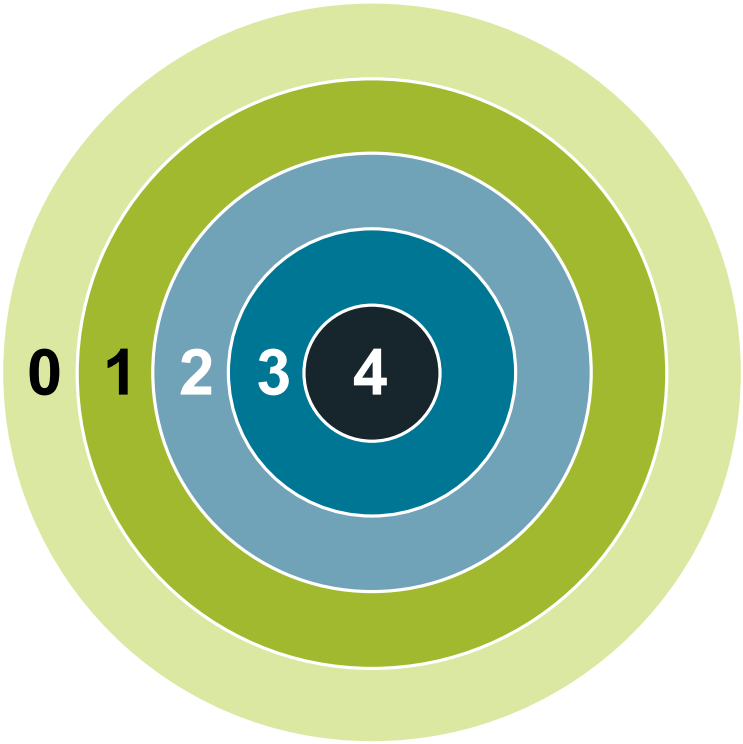
Bit	Name	Reset	Access	Description
31:0	OC	0x0	RW	**Output Compare Value**
	Shows the Compare Value for the channel			

15.5.15 RTCC_CCx_ICVALUE - Input Capture Value Register

Offset	Bit Position																														
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IC																															

Bit	Name	Reset	Access	Description
31:0	IC	0x0	R	**Input Capture Value**
	Shows the Capture Value for the channel			

16. BURTC - Back-Up Real Time Counter



Quick Facts

What?

The BURTC is a 32 bit counter which operates on a low frequency oscillator, and is capable of running in all Energy Modes.

Why?

It can provide periodic Wakeup events and PRS signals which can be used to wake up peripherals from any energy mode.

The availability of the BURTC in EM4, where most of the device is powered down, makes it ideal for keeping track of time in EM4.

How?

The BURTC provides a very wide range of periods for the interrupts facilitating flexible ultra-low energy operation.

16.1 Introduction

The Back-Up Real Time Counter (BURTC) is a 32-bit counter which operates on a low frequency oscillator, and is capable of running in all Energy Modes. It can provide periodic Wakeup events and PRS signals which can be used to wake up peripherals from any energy mode. The BURTC provides a very wide range of periods for the interrupts facilitating flexible ultra-low energy operation. The availability of the BURTC in EM4, where most of the device is powered down, makes it ideal for keeping track of time in EM4. A single compare channel is available which can be used to trigger an interrupt and/or wake the device up from a low energy mode.

16.2 Features

A low frequency oscillator is used as clock signal and the BURTC with one compare channel which can trigger wake-up, generate PRS signalling, or capture system events. 32-bit resolution and selectable prescaling allows the system to stay in low energy modes for long periods of time and still maintain reliable timekeeping.

- 32-bit Real Time Counter
- 15-bit pre-counter for flexible frequency scaling of main counter
- EM2/3/4 operation and wakeup
- Reset only by External Pin and Power-On Resets
- Interrupt/wake up event after deterministic intervals
- PRS Outputs
- Debug mode
  - Configurable to either run or stop when processor is stopped (break)

16.3 Functional Description

An overview of the BURTC module is shown in Figure 16.1 BURTC Overview on page 435.

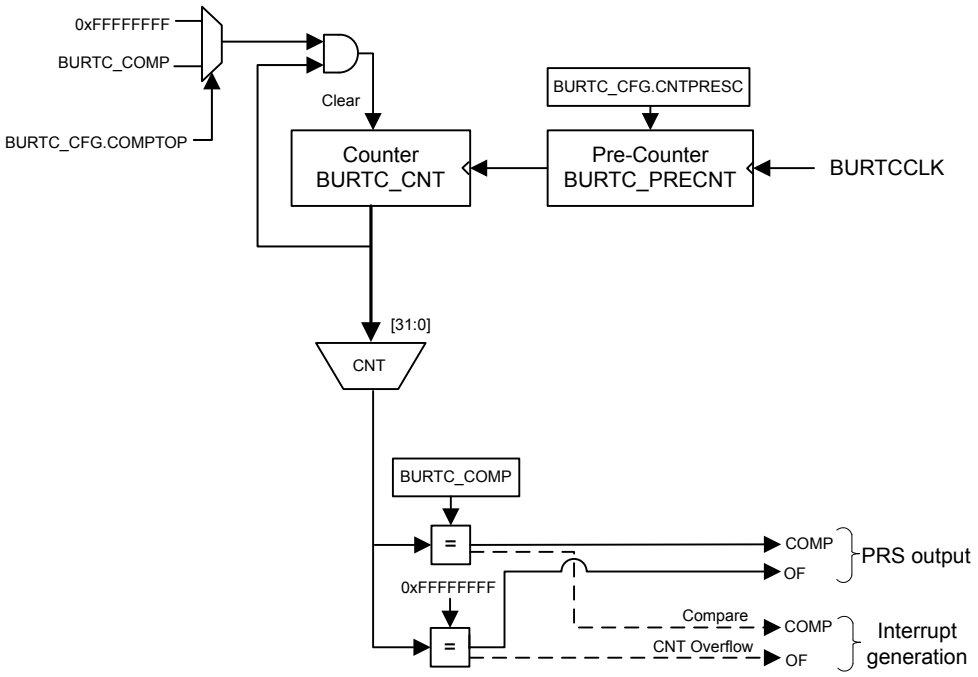


Figure 16.1. BURTC Overview

16.3.1 Clock Selection

The BURTC source clock (BURTCCLK) can be selected to be the LFXO, LFRCO, or ULFRCO by configuring the CMU_EM4GRPACLKCTRL.CLKSEL bitfield. Note that in EM3, only ULFRCO is a valid source clock.

16.3.2 Configuration

To configure and use the BURTC properly, the following programming sequence must be followed:

1. Configure any desired options in the BURTC_CFG register. Note that the BURTC_CFG register can only be written when BURTC_EN.EN = 0 - a bus fault will occur if writing BURTC_CFG register while BURTC_EN.EN = 1.
2. Set BURTC_EN.EN = 1.
3. Set BURTC_CMD.START = 1 to start the BURTC counter.

   Note: All low frequency synchronization registers can only be programmed after EN is set to 1. The BURTC counter will only start to count once START command is issued. For HV Sync registers (e.g., BURTC_CMD), the first bitfield write will occur without issue. However, on subsequent bitfield writes to HV Sync registers, the firmware needs to poll the corresponding bit in BURTC_SYNCBUSY before programming the same bitfield once again.

To stop the BURTC, set BURTC_CMD.STOP = 1

16.3.3 Debug Features and Description

By default, the BURTC is halted when code execution is halted from the debugger. By setting the DEBUGRUN bit in the BURTC_CFG register, the BURTC will continue to run even when the debugger has halted the system.

16.3.4 Counter

The BURTC consists of two counters: the 32-bit main counter, BURTC_CNT, and a 15-bit pre-counter, BURTC_PRECNT. The pre-counter is a free running counter clocked by low frequency clock, used to generate a specific frequency for the main counter. The pre-counter will be counting only when the BURTC_CFG.CNTPRESC value is set greater than 0.

The BURTC peripheral clock is requested by setting the EN bit in BURTC_EN. Then BURTC can be enabled by setting the command register START in BURTC_CMD. When the BURTC is enabled and BURTC_CFG.CNTPRESC > 0, the pre-counter (BURTC_PRECNT) increments upon each positive clock edge of the BURTCCLK, wrapping around to zero when it overflows.

The main counter can be accessed in BURTC_CNT register, and counts at frequency determined by the CNTPRESC bitfiled in BURTC_CFG. Setting CNTPRESC to 0 gives the maximum resolution, with the main counter clocked at the same frequency as the BURTCCLK. When CNTPRESC > 0, the main counter increments upon each tick given from the pre-counter, allowing the main counter ticks to be power-of-2 divisions of the BURTCCLK.

The Table 16.1 BURTC Resolution vs Overflow, $F_{BURTCCLK}$ = 32768 Hz on page 436 table below shows the BURTC Resolution vs Overflow Time when using a 32768 Hz oscillator as the source clock of BURTC.

Table 16.1. BURTC Resolution vs Overflow, $F_{BURTCCLK}$ = 32768 Hz

BURTC_CFG.CNTPRESC	Main counter period, T_{CNT}	Overflow Time
DIV1	30.5 µs	36.4 hours
DIV2	61 µs	72.8 hours
DIV4	122 µs	145.6 hours
DIV8	244 µs	12 days
DIV16	488 µs	24 days
DIV32	977 µs	48 days
DIV64	1.95 ms	97 days
DIV128	3.91 ms	194 days
DIV256	7.81 ms	388 days
DIV512	15.6 ms	776 days
DIV1024	31.25 ms	4.2 years
DIV2048	62.5 ms	8.5 years
DIV4096	0.125 s	17 years
DIV8192	0.25 s	34 years
DIV16384	0.5 s	68 years
DIV32768	1 s	136 years

By default, the counter will keep counting until it reaches the top value, 0xFFFFFFFF, and then it wrap around and continue counting from zero. If COMPTOP in BURTC_CFG is set, the main counter will wrap to 0 on a Compare value match (i.e., BURTC_CNT = BURTC_COMP). If using the Compare value match, make sure to set COMPTOP prior to or at the same time the BURTC is enabled. Setting COMPTOP after enabling the BURTC will result in a bus fault error.

The counters of the BURTC, BURTC_CNT and BURTC_PRECNT, can at any time be written by software, as long as the registers are not locked using BURTC_LOCKKEY. All BURTC control registers with Sync Type HV uses the 2 FF synchronization scheme.

Note: Writing to the BURTC_PRECNT register may alter the frequency of the ticks for the BURTC_CNT register.

16.3.5 Compare Channel

A single compare channel is available in the BURTC. The compare value is set in BURTC_COMP register. If BURTC_CFG.COMPTOP is set, the main counter will clear to 0 when it matches the value set in BURTC_COMP.

16.3.6 Interrupts

The BURTC has 2 interrupts: one for Overflow and another for for Compare match event. Individual interrupts are enabled by BURTC_IEN register bits, and the respective bits can be used as EM2 wakeup. BURTC_EM4WUEN enables the wakeup enable from EM4 for those events.

16.3.7 Register Lock

To prevent accidental writes to the BURTC registers, the BURTC_LOCK register can be written to any other value than the unlock value. To unlock the register, write the unlock value to BURTC_LOCKKEY. Registers affected by this lock are:

- BURTC_CFG
- BURTC_EN
- BURTC_CMD
- BURTC_PRECNT
- BURTC_CNT
- BURTC_COMP
- BURTC_IEN

16.4 BURTC Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	BURTC_IPVERSION	R	IP version ID
0x004	BURTC_EN	RW ENABLE	Module Enable Register
0x008	BURTC_CFG	RW CONFIG	Configuration Register
0x00C	BURTC_CMD	W LFSYNC	Command Register
0x010	BURTC_STATUS	RH	Status Register
0x014	BURTC_IF	RWH INTFLAG	Interrupt Flag Register
0x018	BURTC_IEN	RW	Interrupt Enable Register
0x01C	BURTC_PRECNT	RW LFSYNC	Pre-Counter Value Register
0x020	BURTC_CNT	RW LFSYNC	Counter Value Register
0x024	BURTC_EM4WUEN	RW	EM4 wakeup request Enable Register
0x028	BURTC_SYNCBUSY	RH	Synchronization Busy Register
0x02C	BURTC_LOCK	W	Configuration Lock Register
0x030	BURTC_COMP	RW LFSYNC	Compare Value Register
0x1000	BURTC_IPVERSION_SET	R	IP version ID
0x1004	BURTC_EN_SET	RW ENABLE	Module Enable Register
0x1008	BURTC_CFG_SET	RW CONFIG	Configuration Register
0x100C	BURTC_CMD_SET	W LFSYNC	Command Register
0x1010	BURTC_STATUS_SET	RH	Status Register
0x1014	BURTC_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1018	BURTC_IEN_SET	RW	Interrupt Enable Register
0x101C	BURTC_PRECNT_SET	RW LFSYNC	Pre-Counter Value Register
0x1020	BURTC_CNT_SET	RW LFSYNC	Counter Value Register
0x1024	BURTC_EM4WUEN_SET	RW	EM4 wakeup request Enable Register
0x1028	BURTC_SYNCBUSY_SET	RH	Synchronization Busy Register
0x102C	BURTC_LOCK_SET	W	Configuration Lock Register
0x1030	BURTC_COMP_SET	RW LFSYNC	Compare Value Register
0x2000	BURTC_IPVERSION_CLR	R	IP version ID
0x2004	BURTC_EN_CLR	RW ENABLE	Module Enable Register
0x2008	BURTC_CFG_CLR	RW CONFIG	Configuration Register
0x200C	BURTC_CMD_CLR	W LFSYNC	Command Register
0x2010	BURTC_STATUS_CLR	RH	Status Register
0x2014	BURTC_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2018	BURTC_IEN_CLR	RW	Interrupt Enable Register
0x201C	BURTC_PRECNT_CLR	RW LFSYNC	Pre-Counter Value Register
0x2020	BURTC_CNT_CLR	RW LFSYNC	Counter Value Register

Offset	Name	Type	Description
0x2024	BURTC_EM4WUEN_CLR	RW	EM4 wakeup request Enable Register
0x2028	BURTC_SYNCBUSY_CLR	RH	Synchronization Busy Register
0x202C	BURTC_LOCK_CLR	W	Configuration Lock Register
0x2030	BURTC_COMP_CLR	RW LFSYNC	Compare Value Register
0x3000	BURTC_IPVERSION_TGL	R	IP version ID
0x3004	BURTC_EN_TGL	RW ENABLE	Module Enable Register
0x3008	BURTC_CFG_TGL	RW CONFIG	Configuration Register
0x300C	BURTC_CMD_TGL	W LFSYNC	Command Register
0x3010	BURTC_STATUS_TGL	RH	Status Register
0x3014	BURTC_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3018	BURTC_IEN_TGL	RW	Interrupt Enable Register
0x301C	BURTC_PRECNT_TGL	RW LFSYNC	Pre-Counter Value Register
0x3020	BURTC_CNT_TGL	RW LFSYNC	Counter Value Register
0x3024	BURTC_EM4WUEN_TGL	RW	EM4 wakeup request Enable Register
0x3028	BURTC_SYNCBUSY_TGL	RH	Synchronization Busy Register
0x302C	BURTC_LOCK_TGL	W	Configuration Lock Register
0x3030	BURTC_COMP_TGL	RW LFSYNC	Compare Value Register

16.5 BURTC Register Description

16.5.1 BURTC_IPVERSION - IP version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP Version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

16.5.2 BURTC_EN - Module Enable Register

Offset	Bit Position																														
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**BURTC Enable**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

16.5.3 BURTC_CFG - Configuration Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0						0x0	0x0
Access																									RW						RW	RW
Name																									CNTPRESC						COMPTOP	DEBUGRUN

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:4	CNTPRESC	0x0	RW	**Counter prescaler value.** Configure counting frequency of the CNT register

Value	Mode	Description
0	DIV1	CLK_CNT = (BURTC LF CLK)/1
1	DIV2	CLK_CNT = (BURTC LF CLK)/2
2	DIV4	CLK_CNT = (BURTC LF CLK)/4
3	DIV8	CLK_CNT = (BURTC LF CLK)/8
4	DIV16	CLK_CNT = (BURTC LF CLK)/16
5	DIV32	CLK_CNT = (BURTC LF CLK)/32
6	DIV64	CLK_CNT = (BURTC LF CLK)/64
7	DIV128	CLK_CNT = (BURTC LF CLK)/128
8	DIV256	CLK_CNT = (BURTC LF CLK)/256
9	DIV512	CLK_CNT = (BURTC LF CLK)/512
10	DIV1024	CLK_CNT = (BURTC LF CLK)/1024
11	DIV2048	CLK_CNT = (BURTC LF CLK)/2048
12	DIV4096	CLK_CNT = (BURTC LF CLK)/4096
13	DIV8192	CLK_CNT = (BURTC LF CLK)/8192
14	DIV16384	CLK_CNT = (BURTC LF CLK)/16384
15	DIV32768	CLK_CNT = (BURTC LF CLK)/32768

3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	COMPTOP	0x0	RW	**Compare Channel is Top Value** When set, the counter is cleared in the clock cycle after a compare match with compare channel

Value	Mode	Description
0	DISABLE	The top value of the BURTC is 4294967295 (0xFFFFFFFF)

Bit	Name	Reset	Access	Description
	1	ENABLE		The top value of the BURTC is given by COMP
0	DEBUGRUN	0x0	RW	**Debug Mode Run Enable**
	Set this bit to enable the BURTC to keep running in debug			

Value	Mode	Description
0	DISABLE	BURTC is frozen in debug mode
1	ENABLE	BURTC is running in debug mode

16.5.4 BURTC_CMD - Command Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															W	W
Name																															STOP	START

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	STOP	0x0	W	**Stop BURTC counter**
	Write a 1 to stop the BURTC counter. Differs from SLOWLFWSYNC behavior in that multiple writes cannot be queued up to same register while EN=0			
0	START	0x0	W	**Start BURTC counter**
	Write a 1 to start the BURTC counter. Differs from SLOWLFWSYNC behavior in that multiple writes cannot be queued up to same register			

16.5.5 BURTC_STATUS - Status Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															R	R
Name																															LOCK	RUNNING

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	LOCK	0x0	R	**Configuration Lock Status**
	Indicates the current status of BURTC Lock			
	Value	Mode		Description
	0	UNLOCKED		All BURTC lockable registers are unlocked.
	1	LOCKED		All BURTC lockable registers are locked.
0	RUNNING	0x0	R	**BURTC running status**
	Indicates the current status of BURTC running			

16.5.6 BURTC_IF - Interrupt Flag Register

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															COMP	OF

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	COMP	0x0	RW	**Compare Match Interrupt Flag**
	Set on a compare match between CNT and COMP.			
0	OF	0x0	RW	**Overflow Interrupt Flag**
	Set on a CNT value overflow.			

16.5.7 BURTC_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															COMP	OF

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	COMP	0x0	RW	**Compare Match Interrupt Flag**
	Set to enable the COMPIF Interrupt			
0	OF	0x0	RW	**Overflow Interrupt Flag**
	Set to enable the OFIF Interrupt			

16.5.8 BURTC_PRECNT - Pre-Counter Value Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																		0x0														
Access																		RW														
Name																		PRECNT														

Bit	Name	Reset	Access	Description
31:15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:0	PRECNT	0x0	RW	**Pre-Counter Value**
	Gives access to the Pre-counter value of the BURTC. Differs from SLOWLFRWSYNC behavior in that multiple writes cannot be queued up to same register while EN=0			

16.5.9 BURTC_CNT - Counter Value Register

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	CNT																															

Bit	Name	Reset	Access	Description
31:0	CNT	0x0	RW	**Counter Value**
	Gives access to the counter value of the BURTC. Differs from SLOWLFRWSYNC behavior in that multiple writes cannot be queued up to same register while EN=0			

16.5.10 BURTC_EM4WUEN - EM4 wakeup request Enable Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															COMPEM4WUEN	OFEM4WUEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	COMPEM4WUEN	0x0	RW	**Compare Match EM4 Wakeup Enable**
	Compare Match EM4 wakeup requests. No Synchronization done into peripheral clock domain.			
0	OFEM4WUEN	0x0	RW	**Overflow EM4 Wakeup Enable**
	Overflow EM4 Wakeup request. No Synchronization done into peripheral clock domain.			

16.5.11 BURTC_SYNCBUSY - Synchronization Busy Register

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											R	R	R	R	R	R
Name																											EN	COMP	CNT	PRECNT	STOP	START

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	EN	0x0	R	**Sync busy for EN**
	Last writing of EN is synchronizing to BURTC clock			
4	COMP	0x0	R	**Sync busy for COMP**
	Last writing of COMP is synchronizing to BURTC clock			
3	CNT	0x0	R	**Sync busy for CNT**
	Last writing of CNT is synchronizing to BURTC clock			
2	PRECNT	0x0	R	**Sync busy for PRECNT**
	Last writing of PRECNT is synchronizing to BURTC clock			
1	STOP	0x0	R	**Sync busy for STOP**
	Last writing of STOP is synchronizing to BURTC clock			
0	START	0x0	R	**Sync busy for START**
	Last writing of START is synchronizing to BURTC clock			

16.5.12 BURTC_LOCK - Configuration Lock Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0xAEE8								
Access																								W								
Name																								LOCKKEY								

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0xAEE8	W	**Configuration Lock Key**

Write any other value than the unlock code to lock BURTC_EN, BURTC_CFG, BURTC_CMD, BURTC_PRECNT, BURTC_CNT and BURTC_COMP registers from editing. Write the unlock code to unlock.

Value	Mode	Description
44776	UNLOCK	Write to unlock all BURTC lockable registers

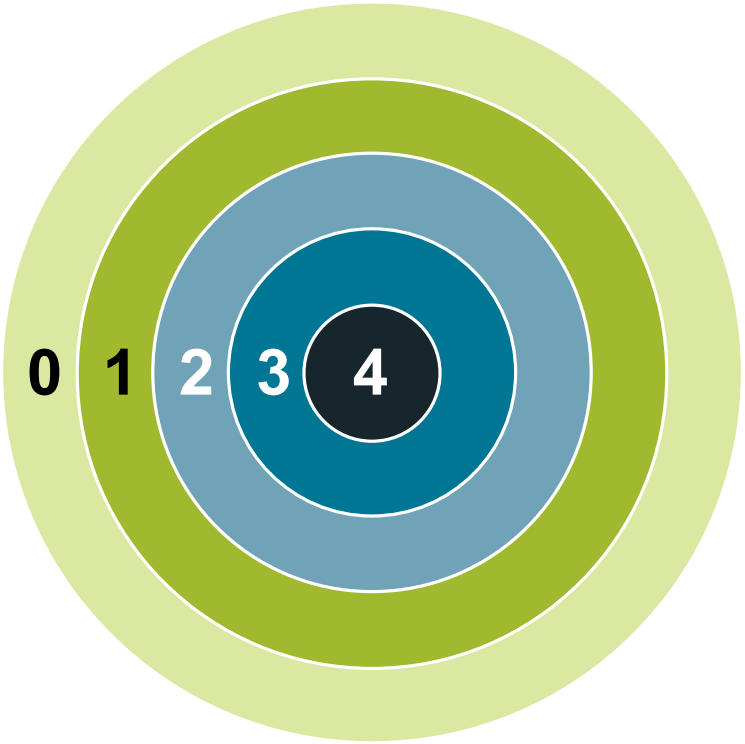
16.5.13 BURTC_COMP - Compare Value Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0																
Access																RW																
Name																COMP																

Bit	Name	Reset	Access	Description
31:0	COMP	0x0	RW	**Compare Value**

A compare match event occurs when CNT is equal to this value. This event sets the COMP interrupt flag. It is also available as a PRS signal. Differs from SLOWLFRWSYNC behavior in that multiple writes cannot be queued up to same register while EN=0

17. BURAM - Backup RAM



Quick Facts

What?

The BURAM is a dedicated 128-byte low-power RAM that is retained in EM4.

Why?

Most of the system, including the RAM, is powered off at EM4 entry to minimize current draw. The purpose of the BURAM is to retain critical data for use when the system wakes up.

How?

Because it is separate from the main system RAM, the BURAM has a dedicated power supply that is not shutdown when the system enters EM4.

17.1 Introduction

The Back-Up RAM (BURAM) is a dedicated 128-byte RAM that remains powered when the system enters EM4. Upon exit from EM4, the data retained in the BURAM can be accessed by the application software.

17.2 Functional Description

The BURAM consists of 32 x 32-bit registers, which are retained in all energy modes, including EM4. Each word in the BURAM is accessible through the corresponding 32 RETx_REG register. Note that each RETx_REG register has an undefined state out of reset.

17.3 BURAM Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	BURAM_RETx_REG	RW	Retention Register
0x1000	BURAM_RETx_REG_SET	RW	Retention Register
0x2000	BURAM_RETx_REG_CLR	RW	Retention Register
0x3000	BURAM_RETx_REG_TGL	RW	Retention Register

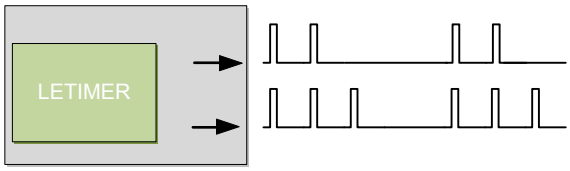
17.4 BURAM Register Description

17.4.1 BURAM_RETx_REG - Retention Register

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	RETREG																															

Bit	Name	Reset	Access	Description
31:0	RETREG	0x0	RW	**Latch based Retention register** The RETREG registers are undefined out of reset. Any written RETREG values will be retained through any event other than a brownout or power-on reset.

18. LETIMER - Low Energy Timer



Quick Facts

What?

The LETIMER is a down-counter that can keep track of time and output configurable waveforms. Running on a 32768 Hz clock, the LETIMER is available in EM0 Active, EM1 Sleep, EM2 DeepSleep, and EM3 Stop.

Why?

The LETIMER can be used to provide repeatable waveforms to external components while remaining in EM2 DeepSleep. It is well suited for applications such as metering systems or to provide more compare values than available in the RTCC.

How?

With buffered repeat and top value registers, the LETIMER can provide glitch-free waveforms at frequencies up to 16 kHz. It can be coupled with RTCC using PRS, allowing advanced time-keeping and wake-up functions in EM2 DeepSleep and EM3 Stop

18.1 Introduction

The LETIMER is a down-counter that can keep track of time and output configurable waveforms with minimal software intervention. Running on a Low Frequency clock, the LETIMER is available in Energy Mode0, Energy Mode 1 and optionally available in Energy Mode 2 and Energy Mode 3. Because of this, it can be used for timing and output generation when most of the device is powered down, allowing simple tasks to be performed while the power consumption of the system is kept at an absolute minimum. It is well suited for applications such as metering systems or to provide more compare values than available in the RTCC. With buffered repeat and top value registers, the LETIMER can provide glitch-free waveforms at frequencies up to 16 kHz. It can be coupled with other peripherals using PRS, allowing advanced time-keeping and wake-up functions

18.2 Features

High-level features

- 24-bit Down counter
- 8-bit prescalar
- 2 Compare match registers
- TOP register can be Timer top value
- TOP register can be double buffered using TOPBUFF register
- Double buffered 8-bit Repeat Register
- Timer Start/Stop/Clear trigger can be from PRS or Software
- Configurable 2 Output pins - Toggle/Pulse/PWM
- Interrupt - Compare match/Timer underflow/Repeat done
- Optionally runs during debug
- 2 output pins can optionally be configured to provide different waveforms on timer underflow:
  - Toggle output pin
  - Pulse output with width of One Prescaled clock period
  - PWM
- 2 PRS Output

18.3 Functional Description

An overview of the LETIMER module is shown in Figure 18.1 LETIMER Overview on page 451. The LETIMER is a 24-bit down-counter with two compare registers, LETIMERn_COMP0 and LETIMERn_COMP1. The LETIMERn_TOP register can optionally act as a top value for the counter. The repeat counter LETIMERn_REP0 allows the timer to count a specified number of times before it stops. Both the LETIMERn_TOP and LETIMERn_REP0 registers can be double buffered by the LETIMERn_TOPBUFF and LETIMERn_REP1 registers to allow continuous operation. The timer can generate a single pin output, or two linked outputs.

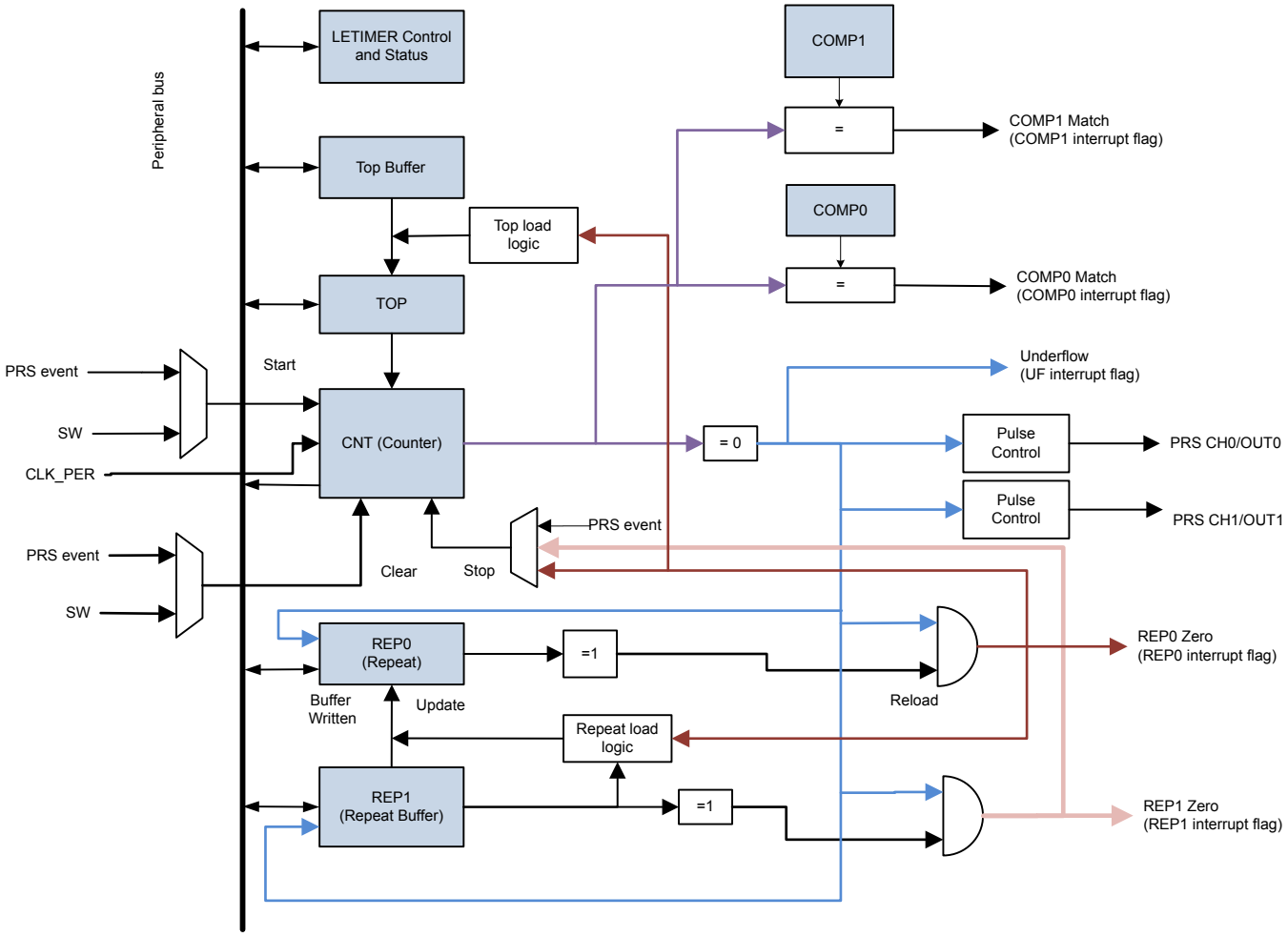


Figure 18.1. LETIMER Overview

18.3.1 Internal Overview

Timer

The timer value can be read using the LETIMERn_CNT register. The value can be written, and it can also be cleared by setting the CLEAR command bit in LETIMERn_CMD. If the CLEAR and START commands are issued at the same time, the timer will be cleared, then start counting at the top value.

Compare Registers

- The LETIMER has two compare match registers, LETIMERn_COMP0 and LETIMERn_COMP1. Each of these compare registers are capable of generating an interrupt when the counter value LETIMERn_CNT is equal to their value. When LETIMERn_CNT is equal to the value of LETIMERn_COMP0, the interrupt flag COMP0 in LETIMERn_IF is set, and when LETIMERn_CNT is equal to the value of LETIMERn_COMP1, the interrupt flag COMP1 in LETIMERn_IF is set.
- **Top Value**

  If CNTTOPEN in LETIMERn_CTRL is set, the value of LETIMERn_TOP acts as the top value of the timer, and LETIMERn_TOP is loaded into LETIMERn_CNT on timer underflow. If CNTTOPEN is cleared to 0, the timer wraps around to 0xFFFFFF. The underflow interrupt flag UF in LETIMERn_IF is set when the timer reaches zero.
- **Repeat Modes**

  By default, the timer wraps around to the top value or 0xFFFFFF on each underflow, and continues counting. The repeat counters can be used to get more control of the operation of the timer, including defining the number of times the counter should wrap around. Four different repeat modes are available, see Table 18.1 LETIMER Repeat Modes on page 452.

Table 18.1. LETIMER Repeat Modes

REPMODE	Mode	Description
0b00	Free-running	The timer runs until it is stopped.
0b01	One-shot	The timer runs as long as LETIMERn_REP0 != 0. LETIMERn_REP0 is decremented at each timer underflow.
0b10	Buffered	The timer runs as long as LETIMERn_REP0 != 0. LETIMERn_REP0 is decremented on each timer underflow. If LETIMERn_REP1 has been written with Non zero value, then it is loaded into LETIMERn_REP0 when LETIMERn_REP0 is about to be decremented to 0 and Timer countinue counting with new LETIMERn_REP0.
0b11	Double	The timer runs as long as LETIMERn_REP0 != 0 or LETIMERn_REP1 != 0. Both LETIMERn_REP0 and LETIMERn_REP1 are decremented at each timer underflow.

  The interrupt flags REP0 and REP1 in LETIMERn_IF are set whenever LETIMERn_REP0 or LETIMERn_REP1 are decremented to 0 respectively. REP0 is also set when the value of LETIMERn_REP1 is loaded into LETIMERn_REP0 in buffered mode.

  Write operations to LETIMERn_REP0 have priority over buffer loads from LETIMERn_REP1.
- **Buffered Top Value**

  In Buffered Mode, If BUFTOP in LETIMERn_CTRL is set, the value of LETIMERn_TOP is buffered by LETIMERn_TOPBUFF. In this mode, the value of LETIMERn_TOPBUFF is loaded into LETIMERn_TOP every time LETIMERn_REP0 is about to decrement to 0. This can be used to generate continually changing output waveforms.

  Write operations to LETIMERn_TOP have priority over buffer loads from LETIMERn__TOPBUFF.

18.3.2 Free Running Mode

In free-running mode, the LETIMER acts as a regular timer and the repeat operation is disabled. When started, the timer runs until it is stopped using the STOP command bit in LETIMERn_CMD/PRS. A state machine for this mode is shown in Figure 18.2 LETIMER State Machine for Free-running Mode on page 453 .

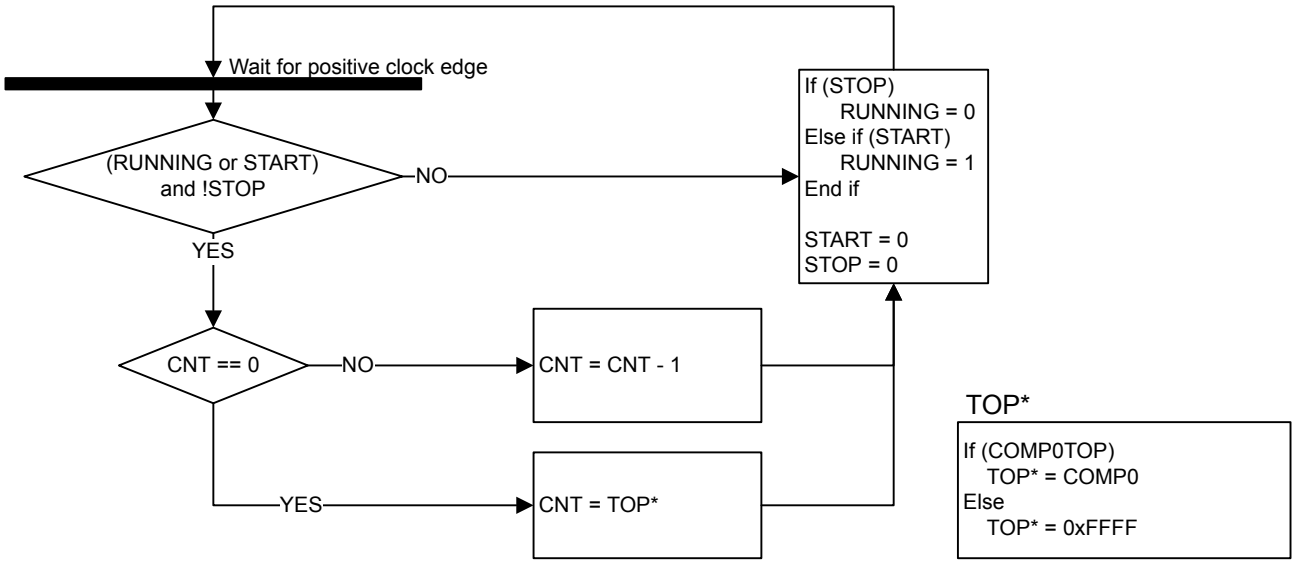


Figure 18.2. LETIMER State Machine for Free-running Mode

Note that the CLEAR command bit in LETIMERn_CMD always has priority over Decrement and Load TOP to LETIMERn_CNT. When the clear command is used, LETIMERn_CNT is set to 0 and an underflow event will not be generated when LETIMERn_CNT wraps around to the top value or 0xFFFFFF. Since no underflow event is generated, no output action is performed. LETIMERn_REP0, LETIMERn_REP1, LETIMERn_COMP0 and LETIMERn_COMP1 are also left untouched.

18.3.3 One-shot Mode

The one-shot repeat mode is the most basic repeat mode. In this mode, the repeat register LETIMERn_REP0 is decremented every time the timer underflows, and the timer stops when LETIMERn_REP0 goes from 1 to 0. In this mode, the timer counts down LETIMERn_REP0 times, i.e. the timer underflows LETIMERn_REP0 times.

Note: Note that write operations to LETIMERn_REP0 have priority over the timer decrement event. If LETIMERn_REP0 is assigned a new value in the same cycle as a timer decrement event occurs, the timer decrement will not occur and the new value is assigned.

LETIMERn_REP0 can be written while the timer is running to allow the timer to run for longer periods at a time without stopping. Write to LETIMERn_REP0 should be done after checking SYNC busy statusFigure 18.3 LETIMER One-shot Repeat State Machine on page 454 .

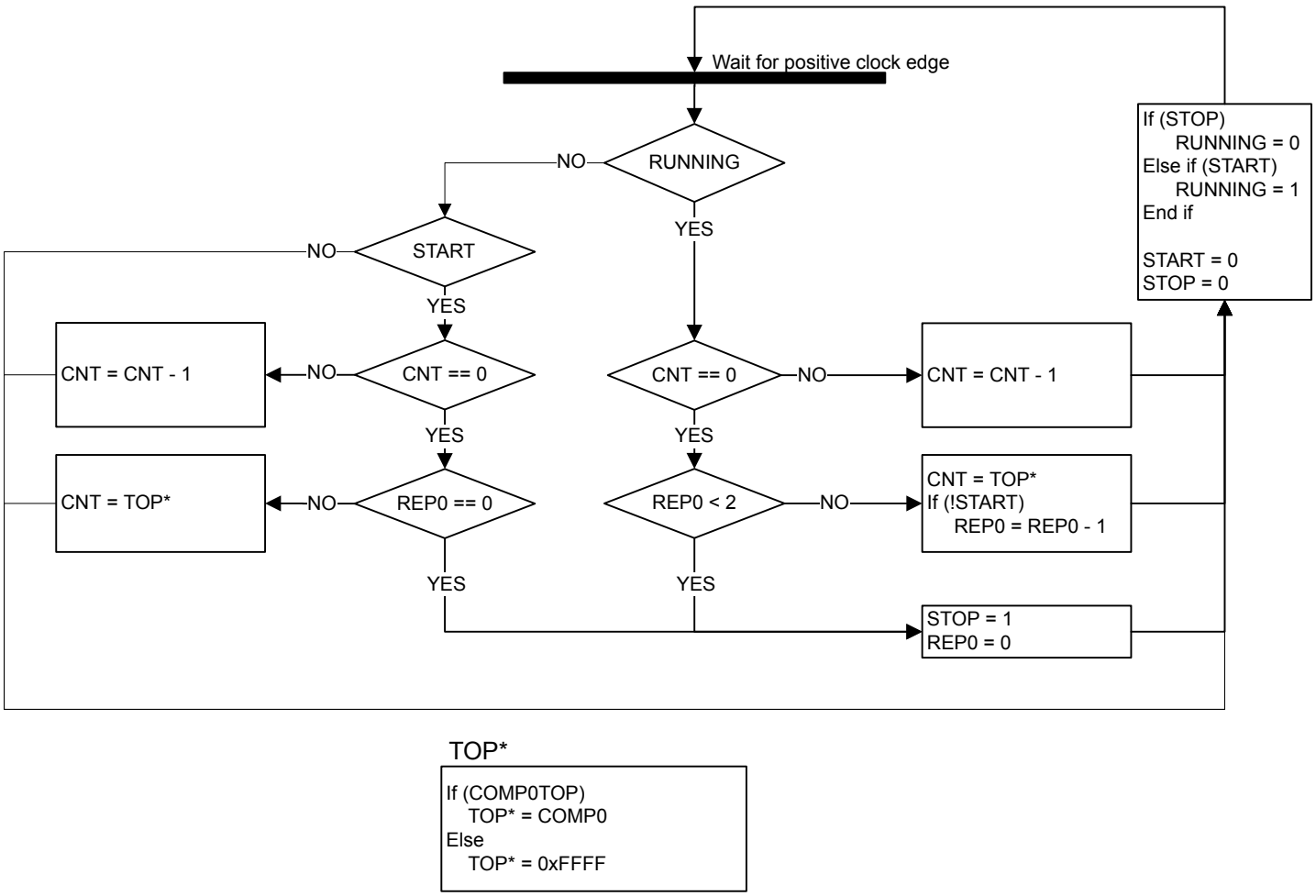


Figure 18.3. LETIMER One-shot Repeat State Machine

18.3.4 Buffered Mode

The Buffered repeat mode allows buffered timer operation. When started, the timer runs LETIMERn_REP0 number of times. If LETIMERn_REP1 has been written since the last time it was used and if it is nonzero, LETIMERn_REP1 is then loaded into LETIMERn_REP0, and counting continues the new number of times. The timer keeps going as long as LETIMERn_REP1 is updated with a nonzero value before LETIMERn_REP0 is finished counting down. The timer top value (LETIMERn_TOP) may also optionally be buffered using Top buff value (LETIMERn_TOPBUFF) by setting BUFTOP in LETIMERn_CTRL.

If the timer is started when both LETIMERn_CNT and LETIMERn_REP0 are zero but LETIMERn_REP1 is non-zero, LETIMERn_REP1 is loaded into LETIMERn_REP0, and the counter counts the loaded number of times.

Used in conjunction with a buffered top value, both the top and repeat values of the timer may be buffered, and the timer can for instance be set to run 4 times with period 7 (top value 6), 6 times with period 200, then 3 times with period 50.

A state machine for the buffered repeat mode is shown in Figure 18.4 LETIMER Buffered Repeat State Machine on page 455. $REP1_{USED}$ shown in the state machine is an internal variable that keeps track of whether the value in LETIMERn_REP1 has been loaded into LETIMERn_REP0 or not. The purpose of this is that a value written to LETIMERn_REP1 should only be counted once. $REP1_{USED}$ is cleared whenever LETIMERn_REP1 is used.

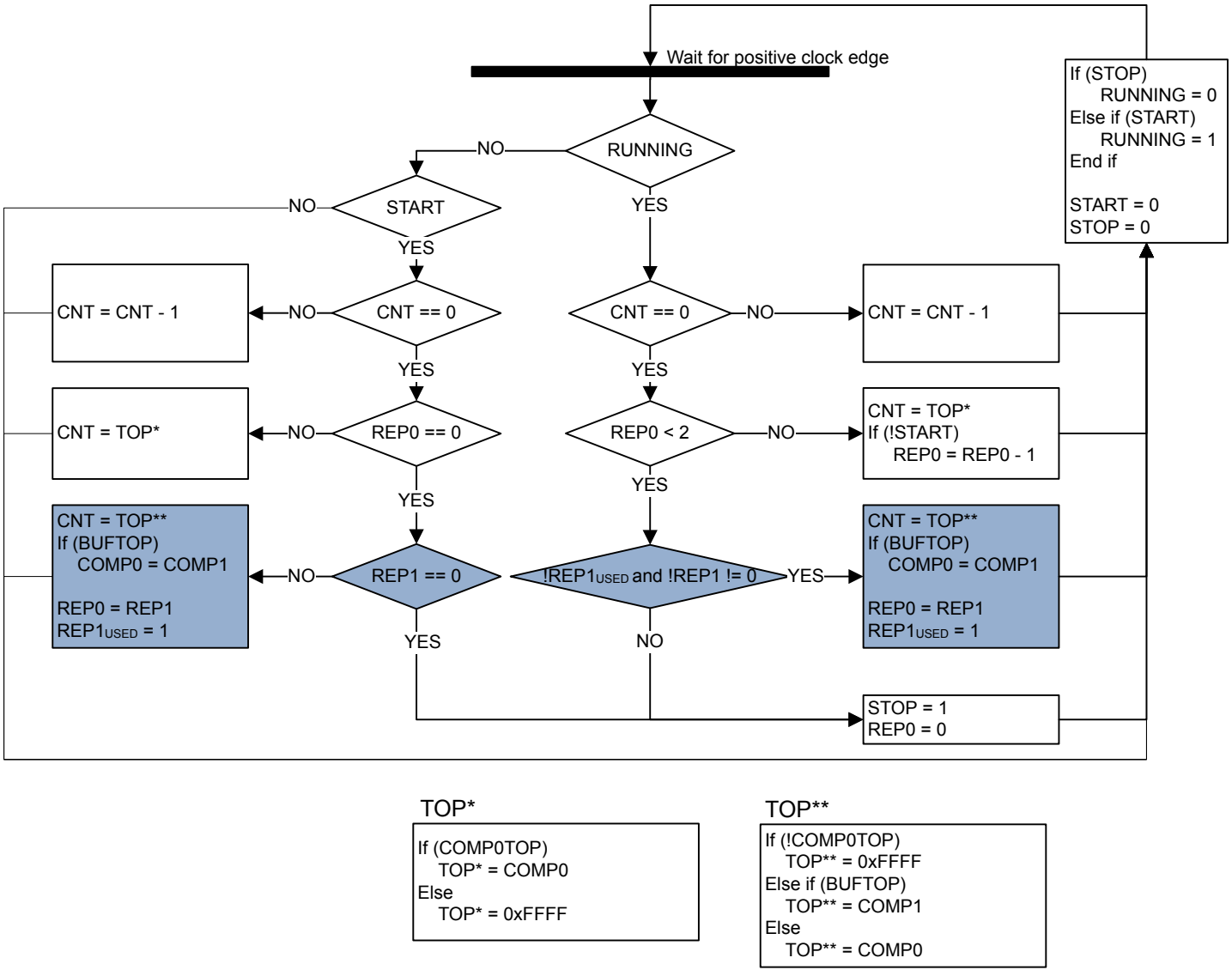


Figure 18.4. LETIMER Buffered Repeat State Machine

18.3.5 Double Mode

The Double repeat mode works much like the one-shot repeat mode. The difference is that, where the one-shot mode counts as long as LETIMERn_REP0 is larger than 0, the double mode counts as long as either LETIMERn_REP0 or LETIMERn_REP1 is larger than 0. As an example, say LETIMERn_REP0 is 3 and LETIMERn_REP1 is 10 when the timer is started. If no further interaction is done with the timer, LETIMERn_REP0 will now be decremented 3 times, and LETIMERn_REP1 will be decremented 10 times. The timer counts a total of 10 times, and LETIMERn_REP0 is 0 after the first three timer underflows and stays at 0. LETIMERn_REP0 and LETIMERn_REP1 can be written at any time. After a write to either of these, the timer is guaranteed to underflow at least the written number of times if the timer is running. Use the Double repeat mode to generate output on both the LETIMER outputs at the same time. The state machine for this repeat mode can be seen in Figure 18.5 LETIMER Double Repeat State Machine on page 456.

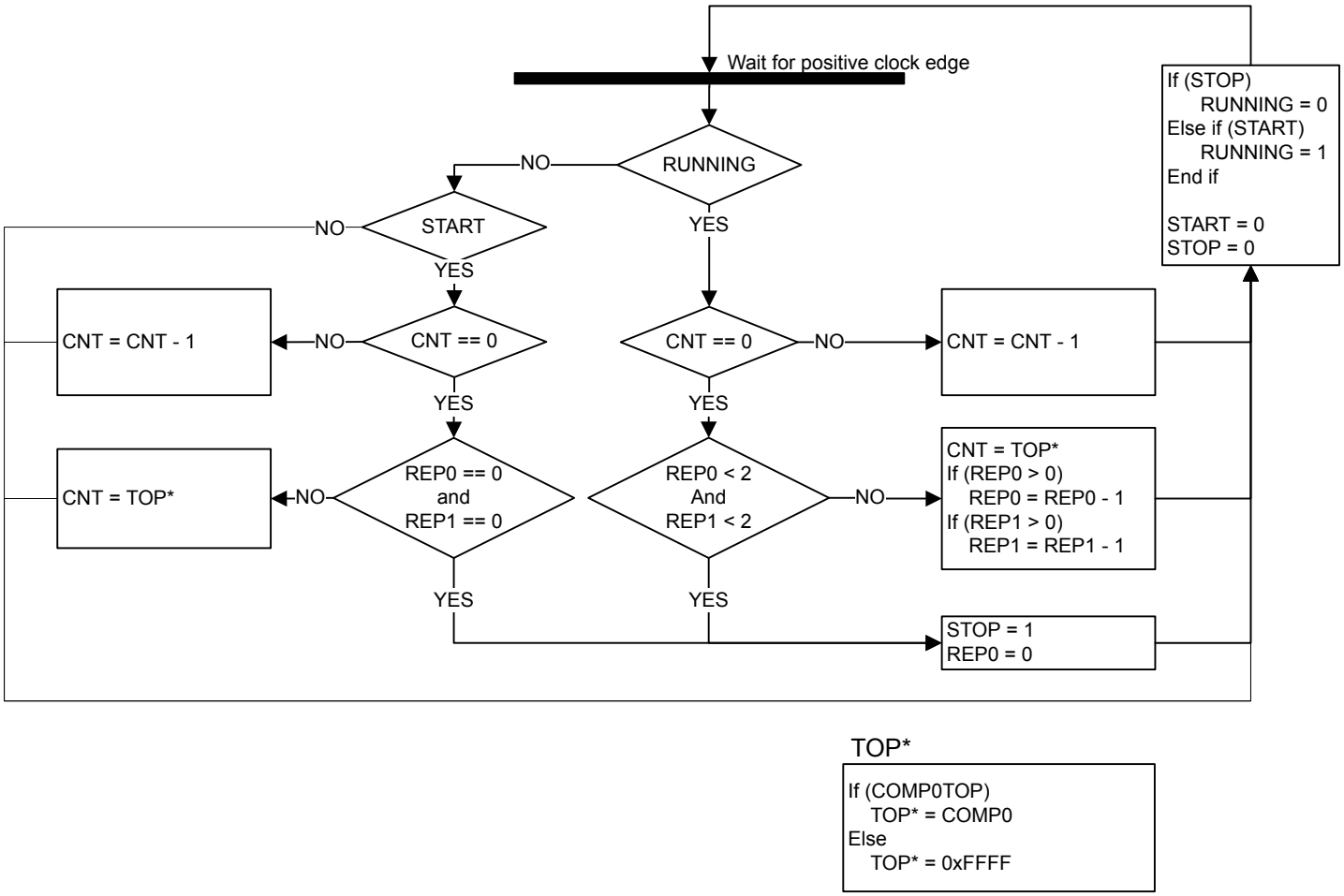


Figure 18.5. LETIMER Double Repeat State Machine

18.4 Clock Frequency

The LETIMER clock source (EM23GRPACLK) is selected in the Clock Management Unit (CMU), and is typically configured to have a frequency of 32 kHz in EM0/1/2 and 1 kHz in EM3. The LETIMER clock prescaler is defined by LETIMERn_CTRL->CNTPRESC.

The LETIMER Prescaled clock frequency is given by Figure 18.6 LETIMER Clock Frequency on page 457.

EM0/1/2 - Clocked by LFRCO

$$f_{LETIMERn\_CLK} = 32768/2^{CNTPRESC}$$

EM3 - Clocked by ULFRCO

$$f_{LETIMERn\_CLK} = 1024/2^{CNTPRESC}$$

Figure 18.6. LETIMER Clock Frequency

The exponent CNTPRESC is a 4 bit value in the LETIMERn_CTRL->CNTPRESC register bits.

To use this module, the LETIMERn_CLK must be enabled by writing 1 to LETIMERn_EN->EN.

18.5 PRS Input Triggers

The LETIMER can be configured to start, stop, and/or clear based on PRS inputs. The diagram showing the functions of the PRS input triggers is shown in Figure 18.7 LETIMER PRS input triggers. on page 458.

There are 3 PRS inputs to the LETIMER, allowing the LETIMER to be started, stopped, or cleared based on the PRS inputs. The PRSSTARTMODE, PRSSTOPMODE, and PRSCLEARMODE bitfields in LETIMERn->PRSMODE select which edge or edge(s) will trigger the start, stop, and/or clear action.

The PRS channel inputs can be configured in the PRS_LETIMER_CLEAR, PRS_LETIMER_START, and PRS_LETIMER_STOP registers in the PRS module.

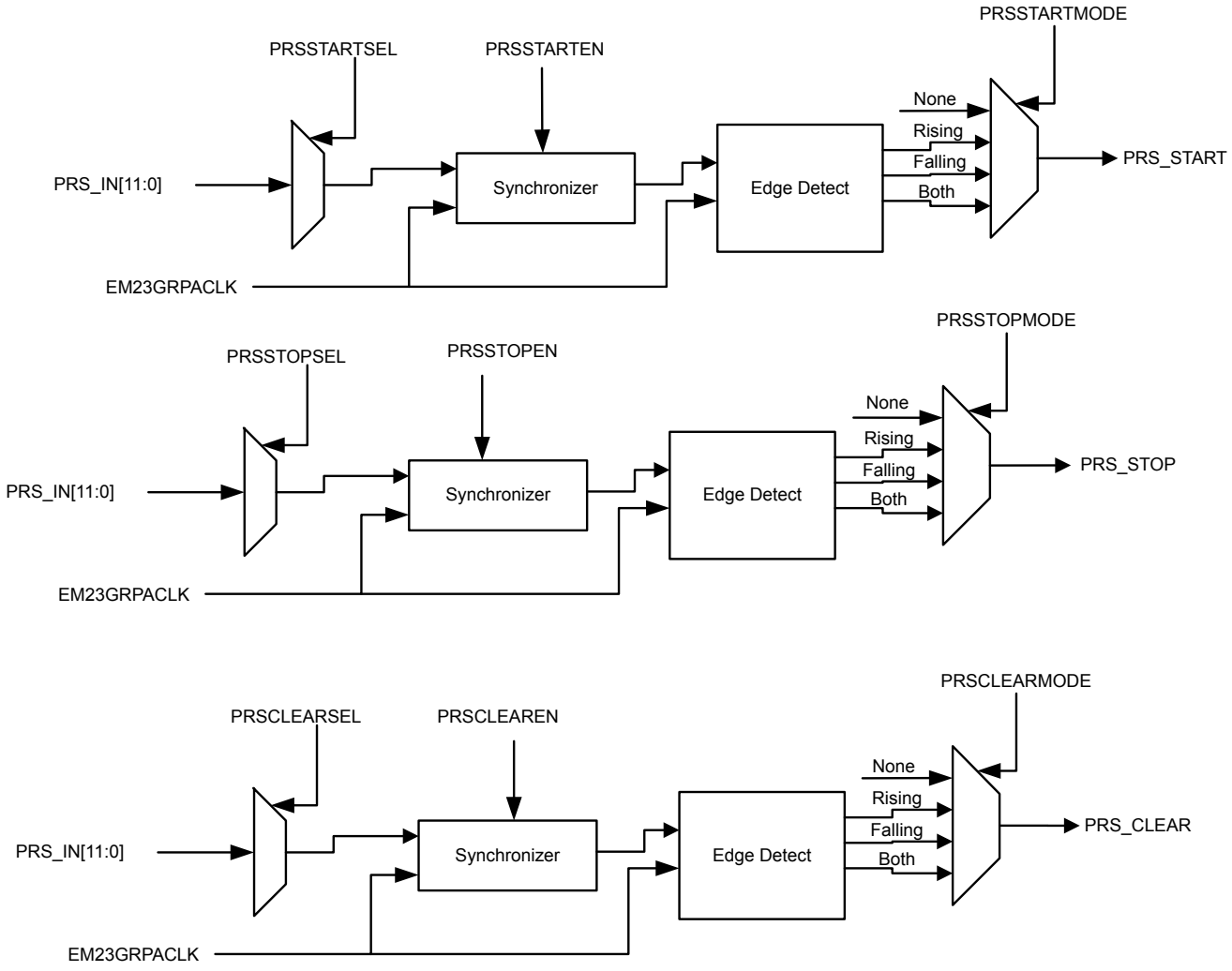


Figure 18.7. LETIMER PRS input triggers.

18.6 Debug

If DEBUGRUN in LETIMERn_CTRL is cleared, the LETIMER automatically stops counting when the CPU is halted during a debug session, and resumes operation when the CPU continues. Because of synchronization, the LETIMER is halted two clock cycles after the CPU is halted, and continues running two clock cycles after the CPU continues. RUNNING in LETIMERn_STATUS is not cleared when the LETIMER stops because of a debug-session.

Set DEBUGRUN in LETIMERn_CTRL to allow the LETIMER to continue counting even when the CPU is halted in debug mode.

18.7 Output Action

For each of the Outputs, an output action can be set.

The output actions can be set by configuring UFOA0 and UFOA1 in LETIMERn_CTRL. UFOA0 defines the action on output 0, while UFOA1 defines the action on output 1. The possible actions are defined in Table 18.2 LETIMER Underflow Output Actions on page 459.

Table 18.2. LETIMER Underflow Output Actions

UF0A0/UF0A1	Mode	Description
0b00	Idle	The output is held at its idle value
0b01	Toggle	The output is toggled on LETIMERn_CNT underflow
0b10	Pulse	The output is held active for one LF clock cycle on LETIMERn_CNT underflow It then returns to its idle value
0b11	PWM	The output is set idle on LETIMERn_CNT underflow and active on compare match with LETIMERn_COMP0/1.

Note: For the Pulse output Disabling LETIMER, Clearing Output while pulse output is generated can affect the pulse width.

Note: For Double mode, OUT0/1 generation is enabled when LETIMERn_REP0/1 != 0 respectively.

The polarity of the outputs can be set individually by configuring OPOL0 and OPOL1 in LETIMERn_CTRL. When these are cleared, their respective outputs have a low idle value and a high active value. When they are set, the idle value is high, and the active value is low. It is recommended to Clear outputs after changing polarity to makesure outputs take their default value.

When using the toggle action, the outputs can be driven to their idle values by setting their respective CTO0/CTO1 command bits in LETIMERn_CTRL. This can be used to put the output in a well-defined state before beginning to generate toggle output, which may be important in some applications. The command bit can also be used while the timer is running.

18.8 PRS Output

The LETIMER outputs can be routed out onto the PRS system. LETn_O0 can be routed to PRS channel 0, and LETn_O1 can be routed to PRS channel 1. Enabling the PRS connection can be done by setting SOURCESEL to LETIMERx and SIGSEL to LETIMERxCHn in PRS_CHx_CTRL.

18.9 Interrupts

The interrupts generated by the LETIMER are combined into one interrupt vector. If the interrupt for the LETIMER is enabled, an interrupt will be made if one or more of the interrupt flags in LETIMERn_IF and their corresponding bits in LETIMER_IEN are set.

18.10 Using the LETIMER in EM3

The LETIMER can be enabled all the way down to EM3 by using the ULFRCO as clock source. This is done by clearing CMU_LFCLKSEL_LFA and setting CMU_LFCLKSEL_LFAE to 1. This will make the RTCC use the internal 1 kHz ultra low frequency RC oscillator (ULFRCO), consuming very little energy. Please note that the ULFRCO is not accurate over temperature and voltage, and it should be verified that the ULFRCO fulfills the timekeeping needs of the application before using this in the design.

18.11 Register access

This module is a Low Energy Peripheral, and supports immediate synchronization. For description regarding immediate synchronization, the reader is referred to the ../../EFR32XG22-RM/EFR32XG22_xrefkeys.dita#xrefs/peripheral_access section.

Since this module is a Low Energy Peripheral, and runs off a clock which is asynchronous to the HFCORECLK, special considerations must be taken when accessing registers.

18.12 Programmer's Model

Important Note : Before writing any LFSYNC register, the module must be enabled (LETIMER_EN->EN) and the LETIMER_SYNCBUSY register should be polled to ensure the SYNC busy of that particular register field is not high.

Write LETIMER Configuration into LETIMER_CTRL Register

Enable clock to LETIMER module by setting LETIMER_EN->EN = 1

If used, write compare values into LETIMER_COMP0 and LETIMER_COMP1

If used, write repeat values into LETIMER_REP0 and LETIMER_REP1

If used, write LETIMER_TOP and LETIMER_TOPBUFF

If PRS is used as a trigger, configure LETIMER_PRSMODE accordingly

Enable Interrupts in LETIMER_IEN

Write LETIMER_CMD register to START Timer

18.12.1 FREE Running Mode

LETIMER operation in Free running Mode with different output modes are shown in Figure 18.8 LETIMER - Free Running Mode Waveform on page 461. In this example, REPMODE in LETIMERn_CTRL is set to FREE, CNTTOPEN also in LETIMERn_CTRL has been set and LETIMERn_TOP has been written to 3. As seen in the figure, LETIMERn_TOP now decides the length of the signal periods. For the toggle mode, the period of the output signal is 2(LETIMERn_TOP + 1), and for the pulse modes, the periods of the output signals are LETIMERn_TOP+1. Note that the pulse outputs are delayed by one period relative to the toggle output. The pulses come at the end of their periods.

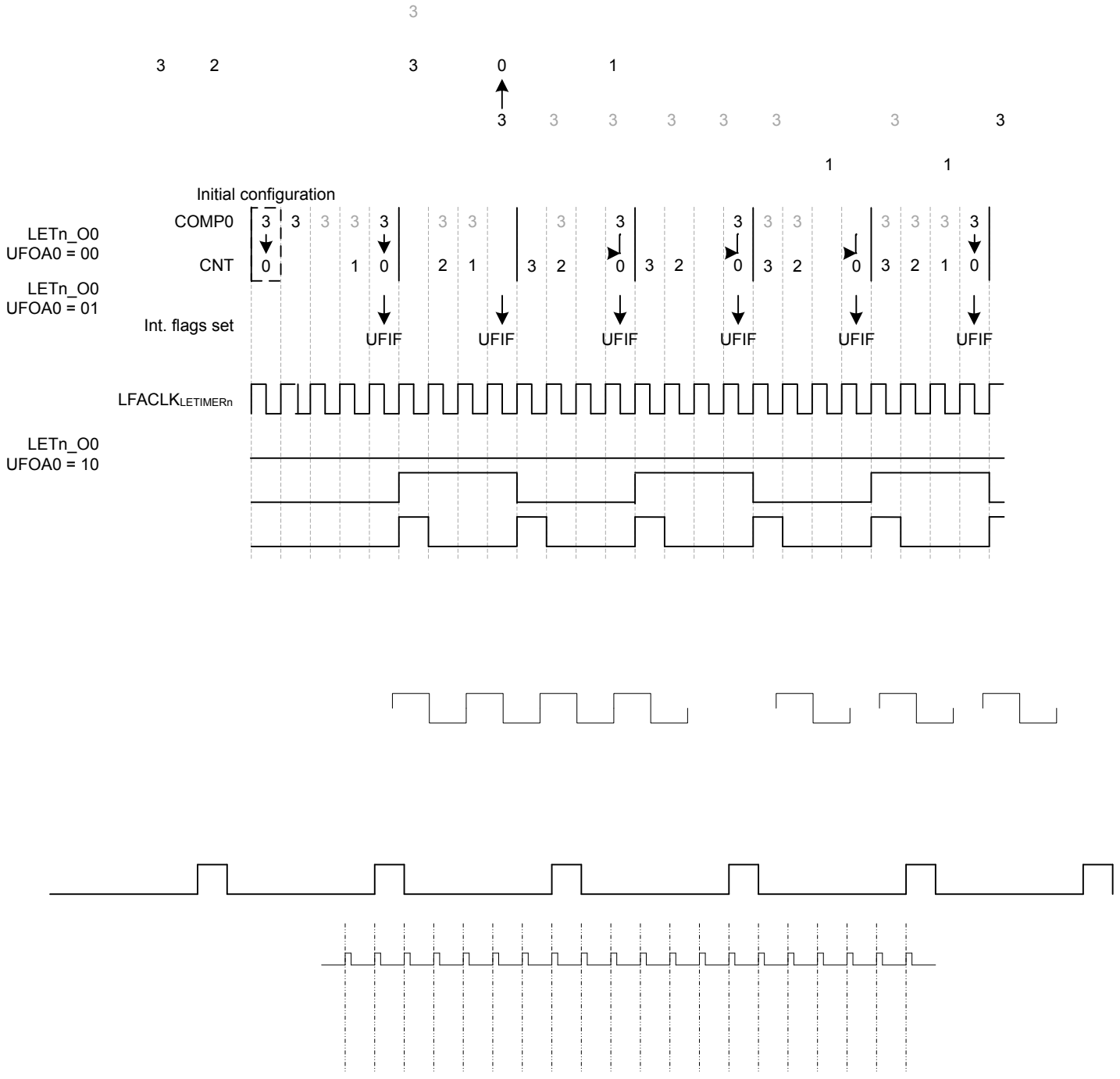


Figure 18.8. LETIMER - Free Running Mode Waveform

18.12.2 One Shot Mode

LETIMER operation in ONESHOT Mode with different output modes are shown in Figure 18.9 LETIMER - One Shot Mode Waveform on page 462. In this example, REPMODE in LETIMERn_CTRL is set to ONESHOT, CNTTOPEN also in LETIMERn_CTRL has been set and LETIMERn_TOP has been written to 3 and LETIMERn_REP0 has been written to 3. The resulting behavior is pretty similar to that shown in Figure 6, but in this case, the timer stops after counting to zero LETIMERn_REP0 times. By using LETIMERn_REP0 the user has full control of the number of pulses/toggles generated on the output.

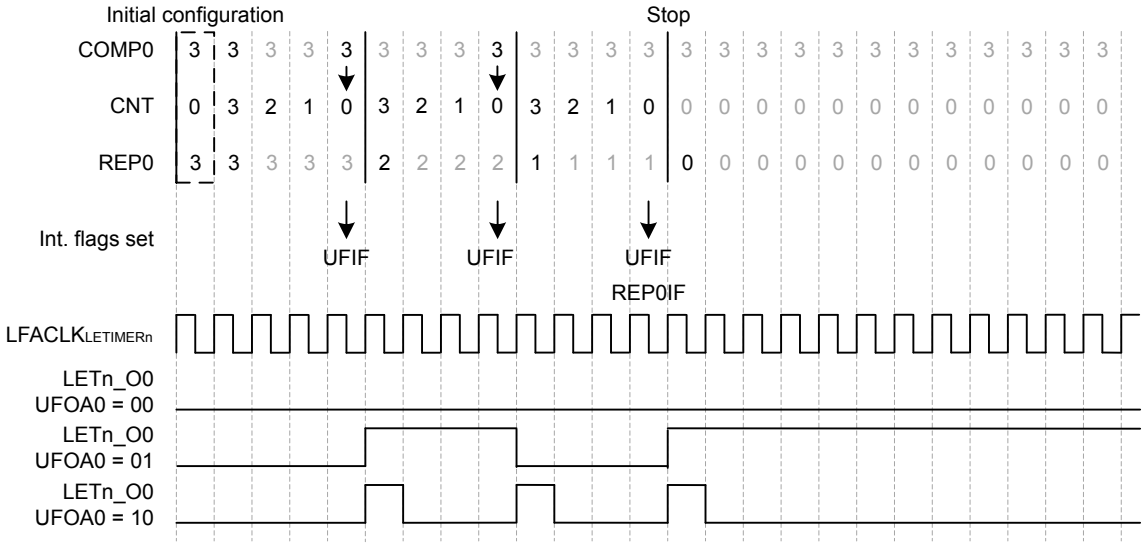


Figure 18.9. LETIMER - One Shot Mode Waveform

18.12.3 DOUBLE Mode

LETIMER operation in DOUBLE Mode with both outputs is shown in Figure 18.10 LETIMER - Double Mode Waveform on page 462. UFOA0 and UFOA1 in LETIMERn_CTRL are configured for pulse output and the outputs are configured for low idle polarity. As seen in the figure, the number written to the repeat registers determine the number of pulses generated on each of the outputs.

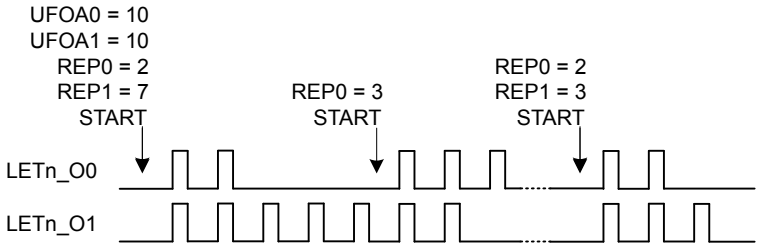


Figure 18.10. LETIMER - Double Mode Waveform

18.12.4 BUFFERED Mode

In BUFFERED Mode LETIMERn_TOPBUFF and LETIMERn_REP1 registers are used as Buffers for LETIMERn_TOP and LETIMERn_REP0 respectiverly. If both LETIMERn_TOP and LETIMERn_REP0 are 0 in buffered mode, and CNTTOPEN and BUFTOP in LETIMERn_CTRL are set, the values of LETIMERn_TOPBUFF and LETIMERn_REP1 are loaded into LETIMERn_TOP and LETIMERn_REP0 respectively when the timer is started. If no additional writes to LETIMERn_REP1 are done before the timer stops, LETIMERn_REP1 determines the number of pulses/toggles generated on the output, and LETIMERn_TOPBUFF determines the period lengths.

As the RTCC can also be used via PRS to start the LETIMER, the RTCC and LETIMER can thus be combined to generate specific pulse-trains at given intervals. Software can update LETIMERn_TOPBUFF and LETIMERn_REP1 to change the number of pulses and pulse-period in each train, but if changes are not required, software does not have to update the registers between each pulse train.

For the example in Figure 18.11 LETIMER - Buffered Mode Waveform on page 463, the initial values cause the LETIMER to generate two pulses with 3 cycle periods, or a single pulse 3 cycles wide every time the LETIMER is started. After the output has been generated, the LETIMER stops, and is ready to be triggered again.

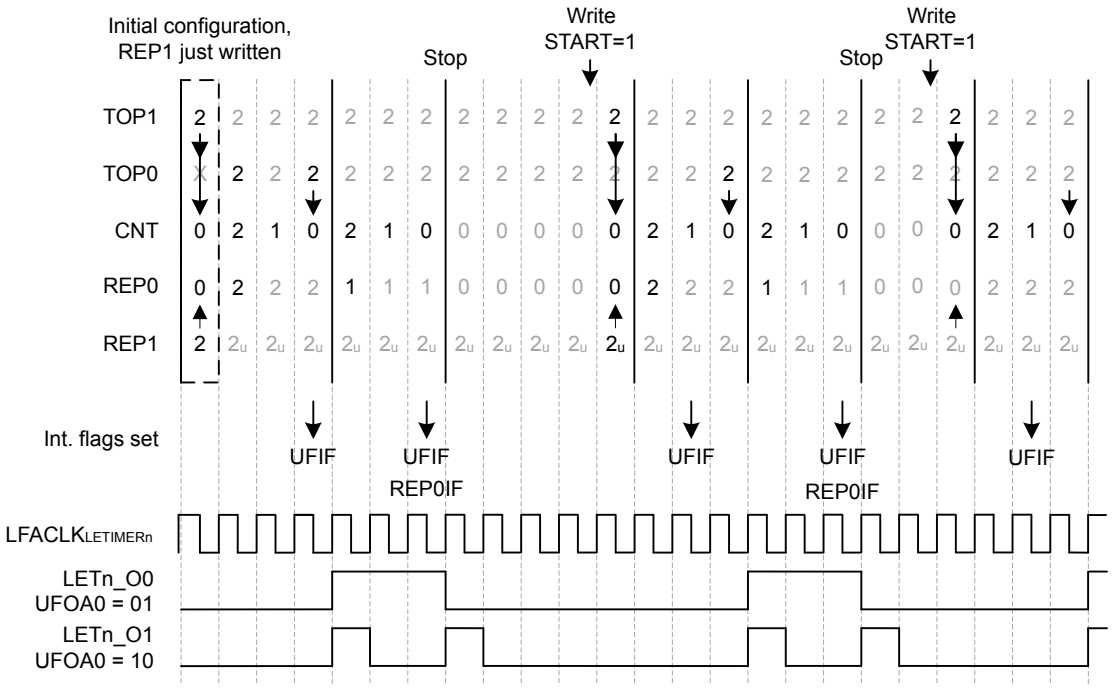


Figure 18.11. LETIMER - Buffered Mode Waveform

18.12.5 Continuous Output Generation

In some scenarios, it might be desired to make LETIMER generate a continuous waveform. Very simple constant waveforms can be generated without the repeat counter as shown in Figure 18.8 LETIMER - Free Running Mode Waveform on page 461, but to generate changing waveforms, using the repeat counter and buffer registers can prove advantageous.

For the example in Figure 18.12 LETIMER - Continuous Operation on page 464, the goal is to produce a pulse train consisting of 3 sequences with the following properties:

- 3 pulses with periods of 3 cycles
- 4 pulses with periods of 2 cycles
- 2 pulses with periods of 3 cycles

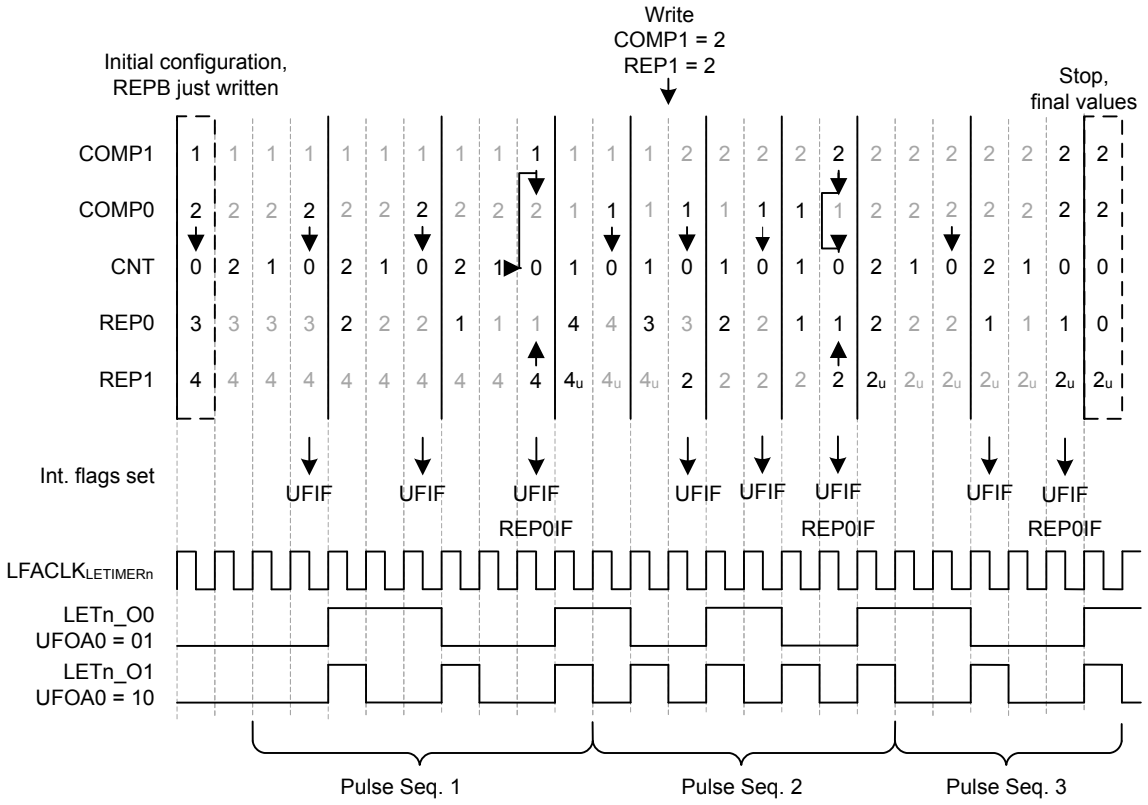


Figure 18.12. LETIMER - Continuous Operation

The first two sequences are loaded into the LETIMER before the timer is started.

LETIMERn_TOP is set to 2 (cycles – 1), and LETIMERn_REP0 is set to 3 for the first sequence, and the second sequence is loaded into the buffer registers, i.e. TOPBUFF is set to 1 and LETIMERn_REP1 is set to 4.

The LETIMER is set to trigger an interrupt when LETIMERn_REP0 is done by setting REP0 in LETIMERn_IEN. This interrupt is a good place to update the values of the buffers. Last but not least REPMODE in LETIMERn_CTRL is set to buffered mode, and the timer is started.

In the interrupt routine the buffers are updated with the values for the third sequence. If this had not been done, the timer would have stopped after the second sequence.

The final result is shown in Figure 18.12 LETIMER - Continuous Operation on page 464. The pulse output is grouped to show which sequence generated which output. Toggle output is also shown in the figure. Note that the toggle output is not aligned with the pulse outputs.

Note: Multiple LETIMER cycles are required to write a value to the LETIMER registers. The example in Figure 18.12 LETIMER - Continuous Operation on page 464 assumes that writes are done in advance so they arrive in the LETIMER as described in the figure.

Figure 18.13 LETIMERn_CNT Not Initialized to 0 on page 465shows an example where the LETIMER is started while LETIMERn_CNT is nonzero. In this case the length of the first repetition is given by the value in LETIMERn_CNT.

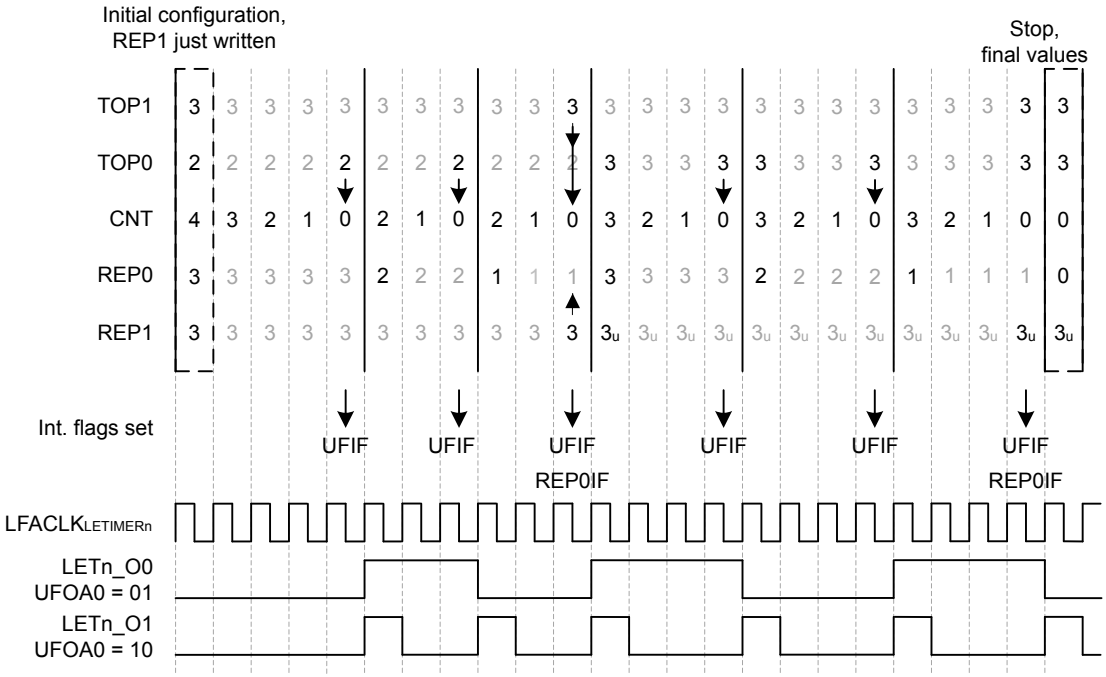


Figure 18.13. LETIMERn_CNT Not Initialized to 0

18.12.6 PWM Output

There are several ways of generating PWM output with the LETIMER, but the most straight-forward way is to use the PWM output mode. This mode is enabled by setting UFOA0 or UFOA1 in LETIMERn_CTRL to 3. In PWM mode, the output is set to idle on timer underflow, and active on LETIMERn_COMP0/1 match, so if for instance CNTTOPEN = 1 and OPOL0 = 0 in LETIMERn_CTRL, LETIMERn_TOP determines the PWM period, and LETIMERn_COMP0/1 determines the active period.

The PWM period in PWM mode is LETIMERn_TOP + 1. There is no special handling of the case where LETIMERn_COMP0/1 > LETIMERn_TOP, so if LETIMERn_COMP0/1 > LETIMERn_TOP, the PWM output is given by the idle output value. This means that for OPOLx = 0 in LETIMERn_CTRL, the PWM output will always be 0 for at least one clock cycle, and for OPOLx = 1 LETIMERn_CTRL, the PWM output will always be 1 for at least one clock cycle.

To generate a PWM signal using the full PWM range, invert OPOLx when LETIMERn_COMP0/1 is set to a value larger than LETIMERn_TOP.

18.13 LETIMER Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	LETIMER_IPVERSION	R	IP version
0x004	LETIMER_EN	RW ENABLE	module en
0x008	LETIMER_CTRL	RW	Control Register
0x00C	LETIMER_CMD	W LFSYNC	Command Register
0x010	LETIMER_STATUS	RH	Status Register
0x018	LETIMER_CNT	RWH LFSYNC	Counter Value Register
0x01C	LETIMER_COMP0	RW	Compare Value Register 0
0x020	LETIMER_COMP1	RW	Compare Value Register 1
0x024	LETIMER_TOP	RWH LFSYNC	Counter TOP Value Register
0x028	LETIMER_TOPBUFF	RW	Buffered Counter TOP Value
0x02C	LETIMER_REP0	RWH LFSYNC	Repeat Counter Register 0
0x030	LETIMER_REP1	RWH LFSYNC	Repeat Counter Register 1
0x034	LETIMER_IF	RWH INTFLAG	Interrupt Flag Register
0x038	LETIMER_IEN	RW	Interrupt Enable Register
0x040	LETIMER_SYNCBUSY	RH	Synchronization Busy Register
0x050	LETIMER_PRSMODE	RW	PRS Input mode select Register
0x1000	LETIMER_IPVERSION_SET	R	IP version
0x1004	LETIMER_EN_SET	RW ENABLE	module en
0x1008	LETIMER_CTRL_SET	RW	Control Register
0x100C	LETIMER_CMD_SET	W LFSYNC	Command Register
0x1010	LETIMER_STATUS_SET	RH	Status Register
0x1018	LETIMER_CNT_SET	RWH LFSYNC	Counter Value Register
0x101C	LETIMER_COMP0_SET	RW	Compare Value Register 0
0x1020	LETIMER_COMP1_SET	RW	Compare Value Register 1
0x1024	LETIMER_TOP_SET	RWH LFSYNC	Counter TOP Value Register
0x1028	LETIMER_TOPBUFF_SET	RW	Buffered Counter TOP Value
0x102C	LETIMER_REP0_SET	RWH LFSYNC	Repeat Counter Register 0
0x1030	LETIMER_REP1_SET	RWH LFSYNC	Repeat Counter Register 1
0x1034	LETIMER_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1038	LETIMER_IEN_SET	RW	Interrupt Enable Register
0x1040	LETIMER_SYNCBUSY_SET	RH	Synchronization Busy Register
0x1050	LETIMER_PRSMODE_SET	RW	PRS Input mode select Register
0x2000	LETIMER_IPVERSION_CLR	R	IP version
0x2004	LETIMER_EN_CLR	RW ENABLE	module en
0x2008	LETIMER_CTRL_CLR	RW	Control Register

Offset	Name	Type	Description
0x200C	LETIMER_CMD_CLR	W LFSYNC	Command Register
0x2010	LETIMER_STATUS_CLR	RH	Status Register
0x2018	LETIMER_CNT_CLR	RWH LFSYNC	Counter Value Register
0x201C	LETIMER_COMP0_CLR	RW	Compare Value Register 0
0x2020	LETIMER_COMP1_CLR	RW	Compare Value Register 1
0x2024	LETIMER_TOP_CLR	RWH LFSYNC	Counter TOP Value Register
0x2028	LETIMER_TOPBUFF_CLR	RW	Buffered Counter TOP Value
0x202C	LETIMER_REP0_CLR	RWH LFSYNC	Repeat Counter Register 0
0x2030	LETIMER_REP1_CLR	RWH LFSYNC	Repeat Counter Register 1
0x2034	LETIMER_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2038	LETIMER_IEN_CLR	RW	Interrupt Enable Register
0x2040	LETIMER_SYNCBUSY_CLR	RH	Synchronization Busy Register
0x2050	LETIMER_PRSMODE_CLR	RW	PRS Input mode select Register
0x3000	LETIMER_IPVERSION_TGL	R	IP version
0x3004	LETIMER_EN_TGL	RW ENABLE	module en
0x3008	LETIMER_CTRL_TGL	RW	Control Register
0x300C	LETIMER_CMD_TGL	W LFSYNC	Command Register
0x3010	LETIMER_STATUS_TGL	RH	Status Register
0x3018	LETIMER_CNT_TGL	RWH LFSYNC	Counter Value Register
0x301C	LETIMER_COMP0_TGL	RW	Compare Value Register 0
0x3020	LETIMER_COMP1_TGL	RW	Compare Value Register 1
0x3024	LETIMER_TOP_TGL	RWH LFSYNC	Counter TOP Value Register
0x3028	LETIMER_TOPBUFF_TGL	RW	Buffered Counter TOP Value
0x302C	LETIMER_REP0_TGL	RWH LFSYNC	Repeat Counter Register 0
0x3030	LETIMER_REP1_TGL	RWH LFSYNC	Repeat Counter Register 1
0x3034	LETIMER_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3038	LETIMER_IEN_TGL	RW	Interrupt Enable Register
0x3040	LETIMER_SYNCBUSY_TGL	RH	Synchronization Busy Register
0x3050	LETIMER_PRSMODE_TGL	RW	PRS Input mode select Register

18.14 LETIMER Register Description

18.14.1 LETIMER_IPVERSION - IP version

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP Version**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

18.14.2 LETIMER_EN - module en

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**module en**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

18.14.3 LETIMER_CTRL - Control Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0							0x0			0x0	0x0	0x0	0x0	0x0		0x0		0x0	
Access													RW							RW			RW	RW	RW	RW	RW		RW		RW	
Name													CNTPRESC							DEBUGRUN			CNTTOPEN	BUFTOP	OPOL1	OPOL0	UFOA1		UFOA0		REPMODE	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	CNTPRESC	0x0	RW	**Counter prescaler value**

Configure counting frequency of the CNT register. - Note - its not recommended to change this setting on the fly.

Value	Mode	Description
0	DIV1	CLK_CNT = (LETIMER LF CLK)/1
1	DIV2	CLK_CNT = (LETIMER LF CLK)/2
2	DIV4	CLK_CNT = (LETIMER LF CLK)/4
3	DIV8	CLK_CNT = (LETIMER LF CLK)/8
4	DIV16	CLK_CNT = (LETIMER LF CLK)/16
5	DIV32	CLK_CNT = (LETIMER LF CLK)/32
6	DIV64	CLK_CNT = (LETIMER LF CLK)/64
7	DIV128	CLK_CNT = (LETIMER LF CLK)/128
8	DIV256	CLK_CNT = (LETIMER LF CLK)/256

15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	DEBUGRUN	0x0	RW	**Debug Mode Run Enable**

Set to keep the LETIMER running in debug mode.

Value	Mode	Description
0	DISABLE	LETIMER is frozen in debug mode
1	ENABLE	LETIMER is running in debug mode

11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	CNTTOPEN	0x0	RW	**Compare Value 0 Is Top Value**

When set, TOP value will be used as Counter Top Value

Value	Mode	Description
0	DISABLE	The top value of the LETIMER is 65535 (0xFFFF)

Bit	Name	Reset	Access	Description
	1	ENABLE		The top value of the LETIMER is given by COMP0
8	BUFTOP	0x0	RW	**Buffered Top**
	Set to load TOPBUFF into TOP when REP0 reaches 0 in BUFFERED mode, allowing a buffered top value.			
	Value	Mode		Description
	0	DISABLE		COMP0 is only written by software
	1	ENABLE		COMP0 is set to COMP1 when REP0 reaches 0
7	OPOL1	0x0	RW	**Output 1 Polarity**
	Defines the idle value of output 1.			
6	OPOL0	0x0	RW	**Output 0 Polarity**
	Defines the idle value of output 0.			
5:4	UFOA1	0x0	RW	**Underflow Output Action 1**
	Defines the action on OUT1 on a LETIMER underflow - IDLE/TOGGLE/PULSE/PWM			
	Value	Mode		Description
	0	NONE		LETIMERn_OUT1 is held at its idle value as defined by OPOL1
	1	TOGGLE		LETIMERn_OUT1 is toggled on CNT underflow
	2	PULSE		LETIMERn_OUT1 is held active for one LETIMER0 clock cycle on CNT underflow. The output then returns to its idle value as defined by OPOL1
	3	PWM		LETIMERn_OUT1 is set idle on CNT underflow, and active on compare match with COMP1
3:2	UFOA0	0x0	RW	**Underflow Output Action 0**
	Defines the action on OUT0 on a LETIMER underflow - IDLE/TOGGLE/PULSE/PWM			
	Value	Mode		Description
	0	NONE		LETIMERn_OUT0 is held at its idle value as defined by OPOL0
	1	TOGGLE		LETIMERn_OUT0 is toggled on CNT underflow
	2	PULSE		LETIMERn_OUT0 is held active for one LETIMER0 clock cycle on CNT underflow. The output then returns to its idle value as defined by OPOL0
	3	PWM		LETIMERn_OUT0 is set idle on CNT underflow, and active on compare match with COMP1
1:0	REPMODE	0x0	RW	**Repeat Mode**
	Repeat Mode - FREE/ONESHOT/BUFFERED/DOUBLE			
	Value	Mode		Description
	0	FREE		When started, the LETIMER counts down until it is stopped by software
	1	ONESHOT		The counter counts REP0 times. When REP0 reaches zero, the counter stops

Bit	Name	Reset	Access	Description
	2	BUFFERED		The counter counts REP0 times. If REP1 has been written, it is loaded into REP0 when REP0 reaches zero, otherwise the counter stops
	3	DOUBLE		Both REP0 and REP1 are decremented when the LETIMER wraps around. The LETIMER counts until both REP0 and REP1 are zero

18.14.4 LETIMER_CMD - Command Register

Offset	Bit Position																														
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												W	W	W	W	W
Name																												CTO1	CTO0	CLEAR	STOP	START

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	CTO1	0x0	W	**Clear Toggle Output 1**
	Set to drive toggle output 1 to its idle value			
3	CTO0	0x0	W	**Clear Toggle Output 0**
	Set to drive toggle output 0 to its idle value			
2	CLEAR	0x0	W	**Clear LETIMER**
	Set to clear LETIMER			
1	STOP	0x0	W	**Stop LETIMER**
	Set to stop LETIMER			
0	START	0x0	W	**Start LETIMER**
	Set to start LETIMER			

18.14.5 LETIMER_STATUS - Status Register

Offset	Bit Position																														
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																R
Name																																RUNNING

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	RUNNING	0x0	R	**LETIMER Running**
	Set when LETIMER is running.			

18.14.6 LETIMER_CNT - Counter Value Register

Offset	Bit Position																														
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0																							
Access									RW																							
Name									CNT																							

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:0	CNT	0x0	RW	**Counter Value**
	Use to read the current value of the LETIMER.			

18.14.7 LETIMER_COMP0 - Compare Value Register 0

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0																						
Access									RW																						
Name									COMP0																						

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:0	COMP0	0x0	RW	**Compare Value 0**
	Compare value for LETIMER.			

18.14.8 LETIMER_COMP1 - Compare Value Register 1

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0																						
Access									RW																						
Name									COMP1																						

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:0	COMP1	0x0	RW	**Compare Value 1**
	Compare and optionally buffered top value for LETIMER.			

18.14.9 LETIMER_TOP - Counter TOP Value Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0																						
Access									RW																						
Name									TOP																						

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:0	TOP	0x0	RW	**Counter TOP Value**
	TOP will be used as Counter TOP Value if CNTTOPEN is set to 1			

18.14.10 LETIMER_TOPBUFF - Buffered Counter TOP Value

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0																						
Access									RW																						
Name									TOPBUFF																						

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:0	TOPBUFF	0x0	RW	**Buffered Counter TOP Value**
	TOPBUFF will be used as Counter TOP Value in BUFFERED Mode if CNTTOPEN and BUFFTOP is set set to 1			

18.14.11 LETIMER_REP0 - Repeat Counter Register 0

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									RW							
Name																									REP0							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	REP0	0x0	RW	**Repeat Counter 0**
	Optional repeat counter.			

18.14.12 LETIMER_REP1 - Repeat Counter Register 1

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									RW							
Name																									REP1							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	REP1	0x0	RW	**Repeat Counter 1**
	Optional repeat counter or buffer for REP0.			

18.14.13 LETIMER_IF - Interrupt Flag Register

Offset	Bit Position																														
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												RW	RW	RW	RW	RW
Name																												REP1	REP0	UF	COMP1	COMP0

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	REP1	0x0	RW	**Repeat Counter 1 Interrupt Flag**
	Set when repeat counter 1 reaches zero.			
3	REP0	0x0	RW	**Repeat Counter 0 Interrupt Flag**
	Set when repeat counter 0 reaches zero or when the REP1 interrupt flag is loaded into the REP0 interrupt flag.			
2	UF	0x0	RW	**Underflow Interrupt Flag**
	Set on LETIMER underflow.			
1	COMP1	0x0	RW	**Compare Match 1 Interrupt Flag**
	Set when LETIMER reaches the value of COMP1.			
0	COMP0	0x0	RW	**Compare Match 0 Interrupt Flag**
	Set when LETIMER reaches the value of COMP0.			

18.14.14 LETIMER_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												RW	RW	RW	RW	RW
Name																												REP1	REP0	UF	COMP1	COMP0

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	REP1	0x0	RW	**Repeat Counter 1 Interrupt Enable**
	Repeat Counter 1 Interrupt Enable			
3	REP0	0x0	RW	**Repeat Counter 0 Interrupt Enable**
	Repeat Counter 0 Interrupt Enable			
2	UF	0x0	RW	**Underflow Interrupt Enable**
	Underflow Interrupt Enable			
1	COMP1	0x0	RW	**Compare Match 1 Interrupt Enable**
	Compare Match 1 Interrupt Enable			
0	COMP0	0x0	RW	**Compare Match 0 Interrupt Enable**
	Compare Match 0 Interrupt Enable			

18.14.15 LETIMER_SYNCBUSY - Synchronization Busy Register

Offset	Bit Position																													
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0		0x0
Access																							R	R	R	R	R	R	R	R		R
Name																							CTO1	CTO0	CLEAR	STOP	START	REP1	REP0	TOP		CNT

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	CTO1	0x0	R	**Sync busy for CTO1**
	Sync busy for CTO1			
8	CTO0	0x0	R	**Sync busy for CTO0**
	Sync busy for CTO0			
7	CLEAR	0x0	R	**Sync busy for CLEAR**
	Sync busy for CLEAR			
6	STOP	0x0	R	**Sync busy for STOP**
	Sync busy for STOP			
5	START	0x0	R	**Sync busy for START**
	Sync busy for START			
4	REP1	0x0	R	**Sync busy for REP1**
	Sync busy for REP1			
3	REP0	0x0	R	**Sync busy for REP0**
	Sync busy for REP0			
2	TOP	0x0	R	**Sync busy for TOP**
	Sync busy for TOP			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CNT	0x0	R	**Sync busy for CNT**
	Sync busy for CNT			

18.14.16 LETIMER_PRSMODE - PRS Input mode select Register

Offset	Bit Position																													
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0				0x0				0x0																			
Access					RW				RW				RW																			
Name					PRSCLEARMODE				PRSSTOPMODE				PRSSTARTMODE																			

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:26	PRSCLEARMODE	0x0	RW	**PRS Clear Mode**
	Mode-NONE/RISING/FALLING/BOTH			

Value	Mode	Description
0	NONE	PRS cannot clear the LETIMER
1	RISING	Rising edge of selected PRS input can clear the LETIMER
2	FALLING	Falling edge of selected PRS input can clear the LETIMER
3	BOTH	Both the rising or falling edge of the selected PRS input can clear the LETIMER

Bit	Name	Reset	Access	Description
25:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:22	PRSSTOPMODE	0x0	RW	**PRS Stop Mode**
	Mode-NONE/RISING/FALLING/BOTH			

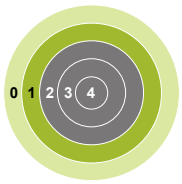
Value	Mode	Description
0	NONE	PRS cannot stop the LETIMER
1	RISING	Rising edge of selected PRS input can stop the LETIMER
2	FALLING	Falling edge of selected PRS input can stop the LETIMER
3	BOTH	Both the rising or falling edge of the selected PRS input can stop the LETIMER

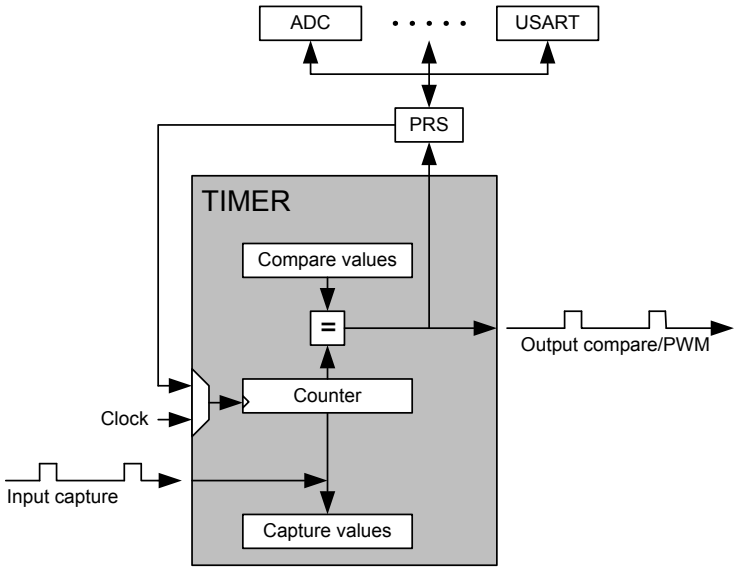
Bit	Name	Reset	Access	Description
21:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:18	PRSSTARTMODE	0x0	RW	**PRS Start Mode**
	Mode-NONE/RISING/FALLING/BOTH			

Value	Mode	Description
0	NONE	PRS cannot start the LETIMER
1	RISING	Rising edge of selected PRS input can start the LETIMER

Bit	Name	Reset	Access	Description
	2	FALLING		Falling edge of selected PRS input can start the LETIMER
	3	BOTH		Both the rising or falling edge of the selected PRS input can start the LETIMER
17:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

19. TIMER - Timer/Counter





Quick Facts

What?

The TIMER (Timer/Counter) keeps track of timing and counts events, generates output waveforms, and triggers timed actions in other peripherals.

Why?

Most applications have activities that need to be timed accurately with as little CPU intervention and energy consumption as possible.

How?

The flexible 16/32-bit timer can be configured to provide PWM waveforms with optional dead-time insertion (e.g. motor control) or work as a frequency generator. The timer can also count events and control other peripherals through the PRS, which offloads the CPU and reduces energy consumption.

19.1 Introduction

The general purpose timer has 3 or 4 compare/capture channels for input capture and compare/Pulse-Width Modulation (PWM) output.

The TIMER module may be 16 or 32 bits wide. Some timers also include a Dead-Time Insertion module suitable for motor control applications.

Refer to the device data sheet to determine the capabilities (capture/compare channel count, width, and DTI) of each timer instance.

19.2 Features

- 16/32-bit auto reload up/down counter
  - Dedicated 16/32-bit reload register which serves as counter maximum
- 3 or 4 Compare/Capture channels
  - Individually configurable as either input capture or output compare/PWM
- Multiple Counter modes
  - Count up
  - Count down
  - Count up/down
  - Quadrature Decoder
  - Direction and count from external pins
- 2x Count Mode
- Counter control from PRS or external pin
  - Start
  - Stop
  - Reload and start
- Inter-Timer connection
  - Allows 32-bit counter mode
  - Start/stop synchronization between several timers
- Input Capture
  - Period measurement
  - Pulse width measurement
  - Two capture registers for each capture channel
    - Capture on either positive or negative edge
    - Capture on both edges
  - Optional digital noise filtering on capture inputs
- Output Compare
  - Compare output toggle/pulse on compare match
  - Immediate update of compare registers
- PWM
  - Up-count PWM
  - Up/down-count PWM
  - Predictable initial PWM output state (configured by SW)
  - Buffered compare register to ensure glitch-free update of compare values
  - Output re-timing to mitigate RF interference
- Clock sources
  - $HFPERCLK_{TIMERn}$
    - 10-bit Prescaler
  - External pin
  - Peripheral Reflex System
- Debug mode
  - Configurable to either run or stop when processor is stopped (halt/breakpoint)
- Interrupts, PRS output and/or DMA request on:
  - Underflow
  - Overflow
  - Compare/Capture event

- Dead-Time Insertion Unit
  - Complementary PWM outputs with programmable dead-time
    - Dead-time is specified independently for rising and falling edge
      - 10-bit prescaler
      - 6-bit time value
    - Outputs have configurable polarity
    - Outputs can be set inactive individually by software.
  - Configurable action on fault
    - Set outputs inactive
    - Clear output
    - Tristate output
  - Individual fault sources
    - One or two PRS signals
    - Debugger
      - Support for automatic restart
    - Core lockup
    - EM2/EM3 entry
  - Configuration lock

19.3 Functional Description

An overview of the TIMER module is shown in Figure 19.1 TIMER Block Overview on page 483 and it consists of a 16/32 bit up/down counter with 3 compare/capture channels connected to pins TIMn_CC0, TIMn_CC1, and TIMn_CC2.

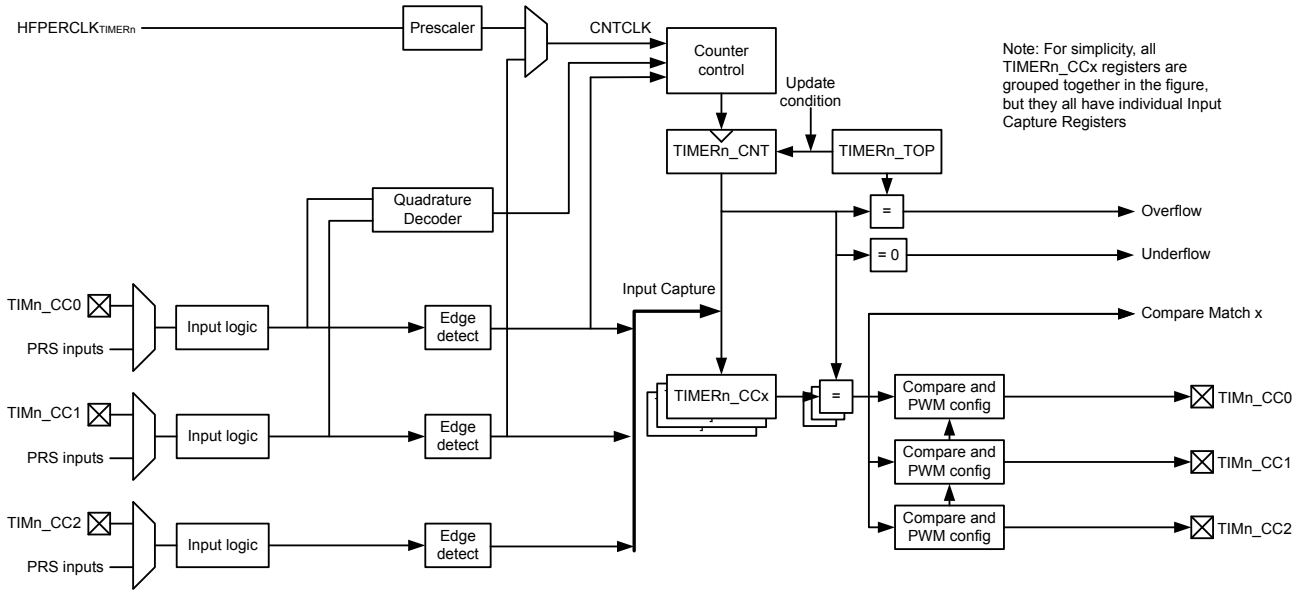


Figure 19.1. TIMER Block Overview

19.3.1 Register Access

The timer module interface consists of multiple register types. Registers of type "RW CONFIG" should only be written when the module is disabled (TIMERn_EN_EN = 0). Registers of type "W SYNC", "R SYNC" or "RW SYNC" should only be read or written when the module is enabled (TIMERn_EN_EN = 1). A typical setup sequence for a TIMER module is as follows:

1. With the TIMER disabled (TIMERn_EN_EN = 0), program any CONFIG registers required for the application.
2. Enable the TIMER by setting EN in TIMERn_EN to 1.
3. Program any non-CONFIG registers required for the application.
4. The TIMER is then ready for use.

19.3.2 Counter Modes

The timer consists of a counter that can be configured to the following modes, using the MODE field in TIMERn_CFG:

- Up-count: Counter counts up until it reaches the value in TIMERn_TOP, where it is reset to 0 before counting up again.
- Down-count: The counter starts at the value in TIMERn_TOP and counts down. When it reaches 0, it is reloaded with the value in TIMERn_TOP.
- Up/Down-count: The counter starts at 0 and counts up. When it reaches the value in TIMERn_TOP, it counts down until it reaches 0 and starts counting up again.
- Quadrature Decoder: Two input channels where one determines the count direction, while the other pin triggers a clock event.

In addition to the TIMER modes listed above, the TIMER also supports a 2x count mode. In this mode the counter increments/decrements by 2 on each clock edge. The 2x count mode can be used to double the PWM frequency when the compare/capture channel is put into PWM mode. The 2x count mode is enabled by setting the X2CNT bitfield in the TIMERn_CTRL register.

The counter value can be read or written by software any time the module is enabled by accessing the CNT field in TIMERn_CNT.

19.3.2.1 Events

The main counter can generate overflow and underflow events during operation.

Overflow (TIMERn_IF_OF) is set when the counter value shifts from TIMERn_TOP to the next value when counting up. In up-count mode and quadrature decoder mode the next value is 0. In up/down-count mode, the next value is TIMERn_TOP-1.

Underflow (TIMERn_IF_UF) is set when the counter value shifts from 0 to the next value when counting down. In down-count mode and quadrature decoder mode, the next value is TIMERn_TOP. In up/down-count mode the next value is 1.

An update event occurs on overflow in up-count mode and on underflow in down-count or up/down count mode. Additionally, an update event also occurs on overflow and underflow in quadrature decoder . This event is used to time updates of buffered values.

19.3.2.2 Operation

Figure 19.2 TIMER Hardware Timer/Counter Control on page 485 shows the hardware timer/counter control. Software can start or stop the counter by setting the START or STOP bits in TIMERn_CMD. The counter value (CNT in TIMERn_CNT) can always be written by software to any 16/32-bit value.

It is also possible to control the counter through either an external pin or PRS input. This is done through the input logic for the compare/capture Channel 0. The timer/counter allows individual actions (start, stop, reload) to be taken for rising and falling input edges. This is configured in the RISEA and FALLA fields in TIMERn_CTRL. The reload value is 0 in up-count and up/down-count mode and TOP in down-count mode.

The RUNNING bit in TIMERn_STATUS indicates if the timer is running or not. If the SYNC bit in TIMERn_CFG is set, the timer is started/stopped/reloaded (external pin or PRS) when any of the other timers are started/stopped/reloaded.

The DIR bit in TIMERn_STATUS indicates the counting direction of the timer at any given time. The counter value can be read or written by software through the CNT field in TIMERn_CNT. In Up/Down-Count mode the count direction will be set to up if the CNT value is written by software.

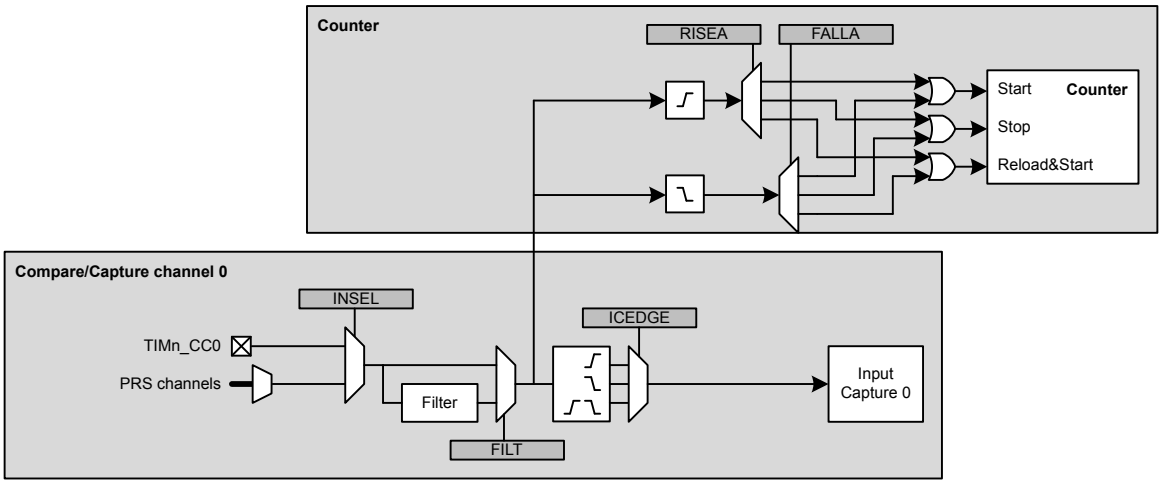


Figure 19.2. TIMER Hardware Timer/Counter Control

19.3.2.3 Clock Source

The counter can be clocked from several sources, which are all synchronized with the incoming peripheral clock for the timer. See Figure 19.3 TIMER Clock Selection on page 485.

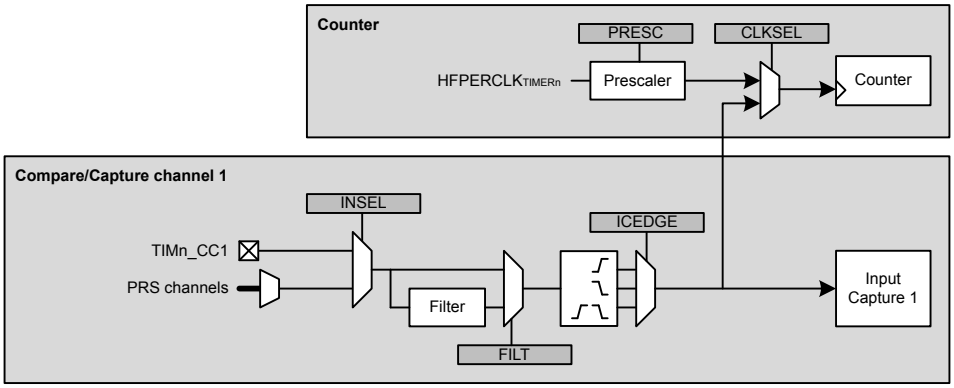


Figure 19.3. TIMER Clock Selection

19.3.2.4 Peripheral Clock

The peripheral clock for the timer ($HFPERCLK_{TIMERn}$) clocks the logic for the timer block, even when it is not the selected clock source.

All TIMER instances in this device family use EM01GRPACLK selected in CMU_EM01GRPACLKCTRL_CLKSEL as their peripheral clock source ($HFPERCLK_{TIMERn}$).

The peripheral clock to each timer can be used as a source with a configurable 10-bit prescaler. The PRESC bitfield in TIMERn_CFG sets the prescaler value, and the incoming peripheral clock will be divided by a factor of (PRESC+1). However, if 2x count mode is enabled and the compare/capture channels are configured for PWM mode, the CC output is updated on both clock edges, so prescaling the peripheral clock will produce an incorrect result. The internal prescale counter is stopped and reset when the timer is stopped.

19.3.2.5 Compare/Capture Channel 1 Input

The timer can also be clocked by positive and/or negative edges on the compare/capture channel 1 input. This input can either come from the TIMn_CC1 pin or one of the PRS channels. The input signal must not have a higher frequency than $f_{HFPERCLK\_TIMERn}/3$ when running from a pin input or a PRS input with FILT enabled in TIMERn_CCx_CFG. When running from PRS without FILT, the frequency can be as high as $f_{HFPERCLK\_TIMERn}$. Note that when clocking the timer from the same pulse that triggers a start (through RISEA/FALLA in TIMERn_CTRL), the starting pulse will not update the counter value.

19.3.2.6 Underflow/Overflow From Neighboring Timer

All timers are linked together (see Figure 19.4 TIMER Connections on page 486), allowing timers to count on overflow/underflow from the lower numbered neighbouring timers to form a larger timer. Note that all timers must be set to count the same direction and less significant timer(s) can only be set to count up or down.

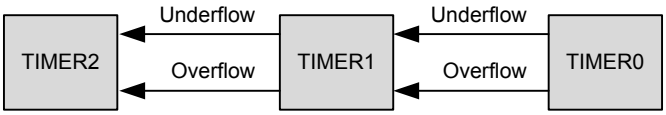


Figure 19.4. TIMER Connections

19.3.2.7 One-Shot Mode

By default, the counter counts continuously until it is stopped. If the OSMEN bit is set in the TIMERn_CFG register, however, the counter is disabled by hardware on the first *update event* (see 19.3.2.1 Events). Note that when the counter is running with CC1 as clock source and OSMEN is set, a CC1 capture event will not take place on the *update event* (CC1 rising edge) that stops the timer.

19.3.2.8 Top Value Buffer

The TIMERn_TOP register can be altered either by writing it directly or by writing to the TIMER_TOPB (buffer) register. When writing to the buffer register the TIMERn_TOPB register will be written to TIMERn_TOP on the next *update event*. Buffering ensures that the TOP value is not set below the actual count value. The TOPBV flag in TIMERn_STATUS indicates whether the TIMERn_TOPB register contains data that has not yet been written to the TIMERn_TOP register (see Figure 19.5 TIMER TOP Value Update Functionality on page 487).

Note: When writing to TIMERn_TOP register directly, the TIMERn_TOPB register value will be invalidated and the TOPBV flag will be cleared. This prevents TIMERn_TOP register from being immediately updated by an existing valid TIMERn_TOPB value during the next *update event*.

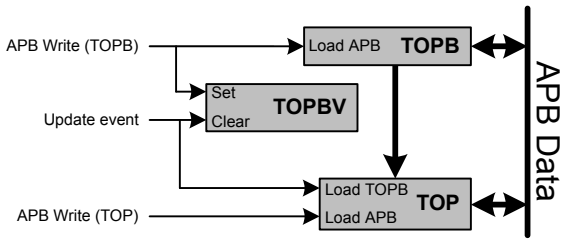


Figure 19.5. TIMER TOP Value Update Functionality

19.3.2.9 Quadrature Decoder

Quadrature decoding mode is used to track motion and determine both rotation direction and position. The quadrature decoder uses two input channels that are 90 degrees out of phase (see Figure 19.6 TIMER Quadrature Encoded Inputs on page 488).

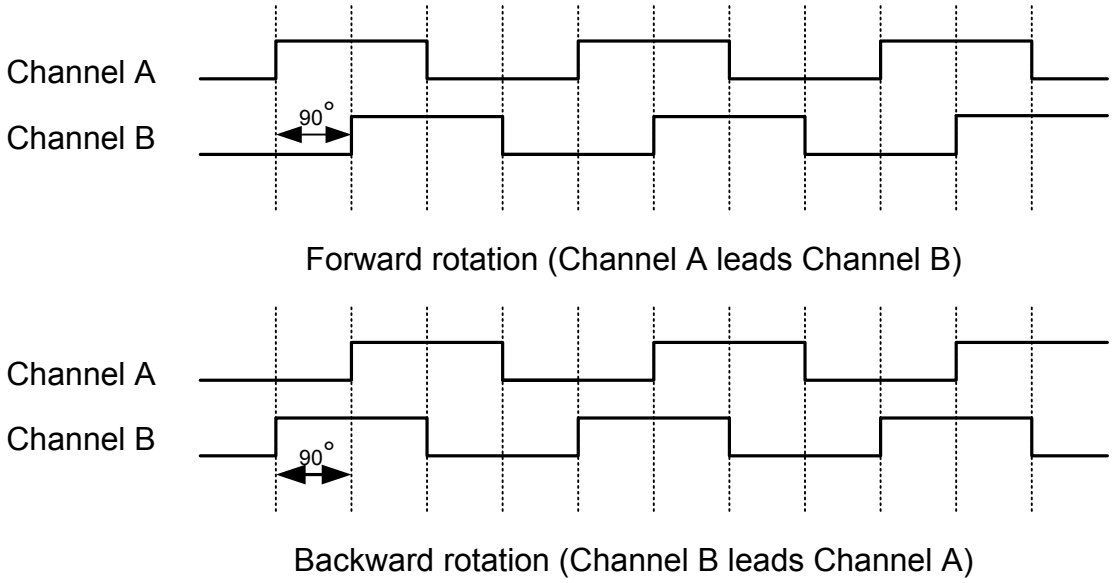


Figure 19.6. TIMER Quadrature Encoded Inputs

In the timer these inputs are tapped from the compare/capture channel 0 (Channel A) and 1 (Channel B) inputs before edge detection. The timer/counter then increments or decrements the counter, based on the phase relation between the two inputs. The DIRCHG flag in TIMERn_IF is set if the count direction changes in quadrature decoder mode. The quadrature decoder supports two channels, but if a third channel (Z-terminal) is available, this can be connected to an external interrupt and trigger a counter reset from the interrupt service routine. By connecting a periodic signal from another timer as input capture on compare/capture Channel 2, it is also possible to calculate speed and acceleration.

Note: In quadrature decoder mode, overflow and underflow triggers an *update event.*

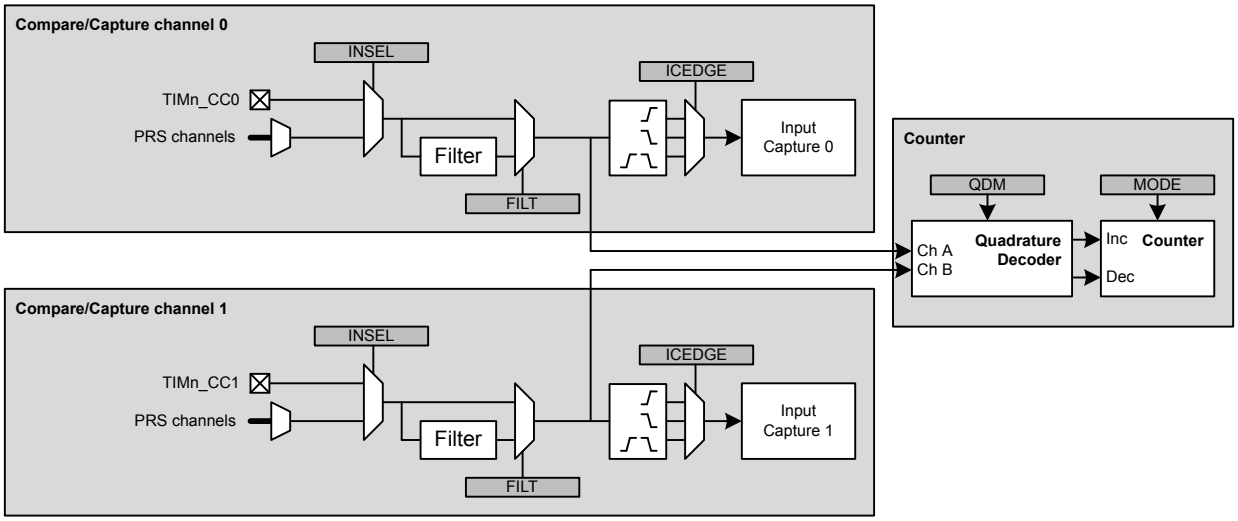


Figure 19.7. TIMER Quadrature Decoder Configuration

The quadrature decoder can be set in either X2 or X4 mode, which is configured in the QDM bit in TIMERn_CFG. See Figure 19.7 TIMER Quadrature Decoder Configuration on page 488

19.3.2.10 X2 Decoding Mode

In X2 Decoding mode, the counter increments or decrements on every edge of Channel A, see Table 19.1 TIMER Counter Response in X2 Decoding Mode on page 489 and Figure 19.8 TIMER X2 Decoding Mode on page 489.

Table 19.1. TIMER Counter Response in X2 Decoding Mode

Channel B	Channel A	
	Rising	Falling
0	Increment	Decrement
1	Decrement	Increment

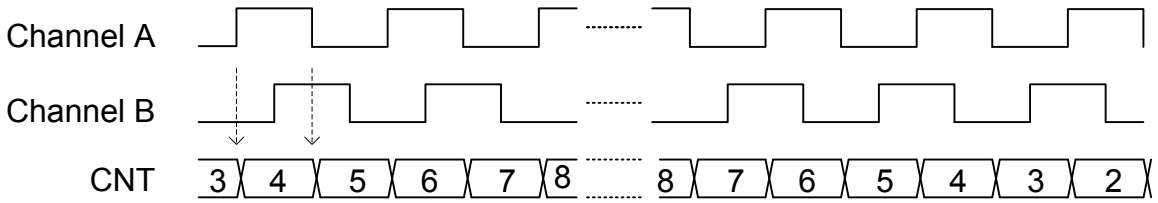


Figure 19.8. TIMER X2 Decoding Mode

19.3.2.11 X4 Decoding Mode

In X4 Decoding mode, the counter increments or decrements on every edge of Channel A and Channel B, see Figure 19.9 TIMER X4 Decoding Mode on page 489 and Table 19.2 TIMER Counter Response in X4 Decoding Mode on page 489.

Table 19.2. TIMER Counter Response in X4 Decoding Mode

Opposite Channel	Channel A		Channel B	
	Rising	Falling	Rising	Falling
Channel A = 0			Decrement	Increment
Channel A = 1			Increment	Decrement
Channel B = 0	Increment	Decrement		
Channel B = 1	Decrement	Increment		

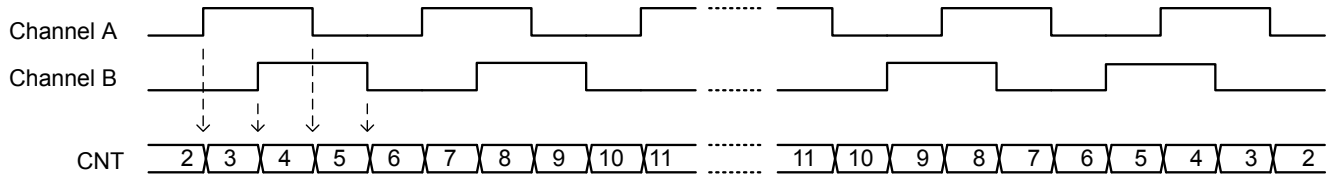


Figure 19.9. TIMER X4 Decoding Mode

19.3.2.12 Rotational Position

To calculate a position Figure 19.10 TIMER Rotational Position Equation on page 490 can be used.

$$pos° = (CNT/X \times N) \times 360°$$

Figure 19.10. TIMER Rotational Position Equation

where X = Encoding type and N = Number of pulses per revolution.

19.3.3 Compare/Capture Channels

The timer contains compare/capture channels, which can be independently configured in the following modes:

1. Input Capture
2. Output Compare
3. PWM

19.3.3.1 Input Pin Logic

Each compare/capture channel can be configured as an input source for the Capture Unit or as external clock source for the timer (see Figure 19.11 TIMER Input Pin Logic on page 490). Compare/capture channels 0 and 1 are the inputs for the quadrature decoder . The input channel can be filtered before it is used, which requires the input to remain stable for up to 5 cycles in a row before the input is propagated to the output.

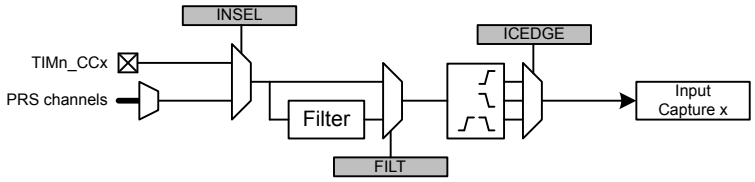


Figure 19.11. TIMER Input Pin Logic

The capture input to the timer may be selected from the dedicated CCx signal for the channel, or a PRS signal. INSEL in TIMERn_CCx_CFG determines the input to the channel. When set to PIN, the selected CCx pin will be used. When INSEL is set to PRSSYNC, a synchronous PRS channel is selected as the source. The synchronous PRS channel is determined by the SPRSSEL field in the PRS_TIMERn_CCx register. Setting INSEL to PRSASYNCLEVEL or PRSASYNCPULSE selects an asynchronous PRS channel as the source. The aynchronous PRS channel is determined by the PRSSEL field in the PRS_TIMERn_CCx register.

The PIN and PRSASYNCLEVEL selections are qualified by a 2-clock input sampler. To recognize and capture the incoming signal, it must be at the new level for at least 2 $HFPERCLK_{TIMERn}$ clock cycles. An additional 5 $HFPERCLK_{TIMERn}$ cycles of filtering can be applied to the signal by enabling the FILT bit in TIMERn_CCx_CFG.

The PRSASYNCPULSE selection can be used to capture higher-speed pulses on an asynchronous PRS input. The input logic for this selection does not qualify the level of the incoming signal. Instead, it will recognize positive or negative edges directly. While the pulse time can be shorter than 1 $HFPERCLK_{TIMERn}$, this mode requires at least 3 $HFPERCLK_{TIMERn}$ clocks between adjacent events. The FILT option is not used in this mode.

Synchronous PRS signals are inherently synchronized to the module clock, and the 2-clock input sampler is not used. However, it is possible to use FILT to enable the 5 $HFPERCLK_{TIMERn}$ filter when using the PRSSYNC option.

19.3.3.2 Compare/Capture Registers

The compare/capture channel registers are prefixed with TIMERn_CCx_, where the x stands for the channel number. Since the compare/capture channels serve three functions (input capture, compare, PWM), different registers are used, depending on the mode the channel is set in.

19.3.3.3 Input Capture

In input capture, the counter value (TIMERn_CNT) can be captured in the Input Capture Register (TIMERn_CCx_ICF) (see Figure 19.12 TIMER Input Capture on page 491). The CCPOL bits in TIMERn_STATUS indicate the polarity of the edge that triggered the capture in TIMERn_CCx_ICF.

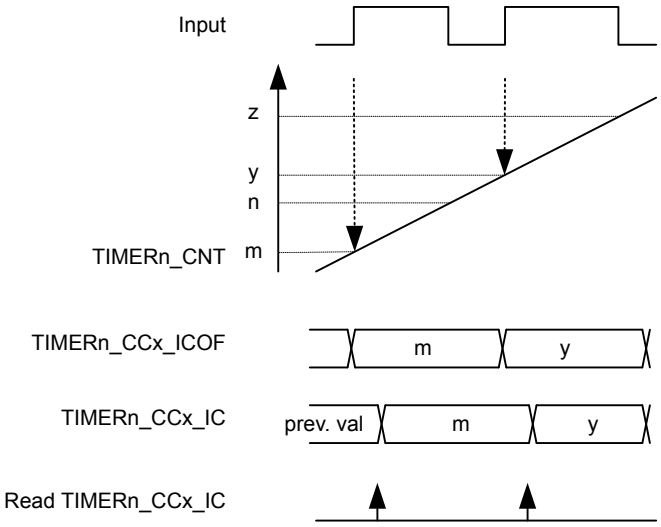


Figure 19.12. TIMER Input Capture

Input captures are buffered into a 2-entry FIFO, allowing 2 subsequent capture events to take place before a read-out is required. Reading TIMERn_CCx_ICF from software or DMA pops the oldest unread value from the FIFO. If TIMERn_CCx_ICF is read when the FIFO is empty (ICFEMPTY in TIMERn_STATUS = 1), the FIFO underflow flag for the channel (ICFUF in TIMERn_IF) will be set. The Input Capture Overflow Register (TIMERn_CCx_ICOF) always contains the newest value in the FIFO. If a new capture is triggered while the FIFO is full, the value in TIMERn_CCx_ICOF will be over-written with the latest value and the FIFO overflow flag (ICFOF in TIMERn_IF) for the channel will be set. Reading TIMERn_CCx_ICOF does not alter the FIFO contents.

The input capture FIFO also has a programmable watermark level that can be configured to generate interrupts or trigger DMA requests when a certain number of empty spots are left in the FIFO. The ICFWLFULL flag inTIMERn_IF will be set when the number of empty spots left in the FIFO is less than or equal to the watermark level programmed in TIMERn_CCx_CFG_ICFWL. At a minimum, a TIMER module will have two FIFO entries, but may have more on future devices.

The ICFEMPTY flag in TIMERn_STATUS indicates when the capture buffer is empty. When this bit reads '0', there is a valid unread capture in the FIFO.

Note: In input capture mode, the timer will only trigger interrupts when it is running.

19.3.3.4 Period/Pulse-Width Capture

Period and/or pulse-width capture can only be possible with Channel 0 (CC0), because this is the only channel that can start and stop the timer. This can be done by setting the RISEA field in TIMERn_CTRL to Clear&Start, and selecting the desired input from either external pin or PRS, see Figure 19.13 TIMER Period and/or Pulse width Capture on page 492. For period capture, the compare/capture channel should then be set to input capture on a rising edge of the same input signal. To capture the width of a high pulse, the compare/capture channel should be set to capture on a falling edge of the input signal. To measure the low pulse-width of a signal, opposite polarities should be chosen.

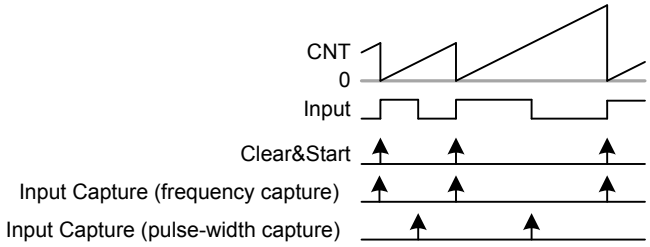


Figure 19.13. TIMER Period and/or Pulse width Capture

19.3.3.5 Compare

Each compare/capture channel contains a comparator which outputs a compare match if the contents of TIMERn_CCx_OC matches the counter value, see Figure 19.14 TIMER Block Diagram Showing Comparison Functionality on page 493. In compare mode, each compare channel can be configured to either set, clear or toggle the output on an event (compare match, overflow or underflow). The output from each channel is represented as an alternative function on the port it is connected to, which needs to be enabled for the CC outputs to propagate to the pins.

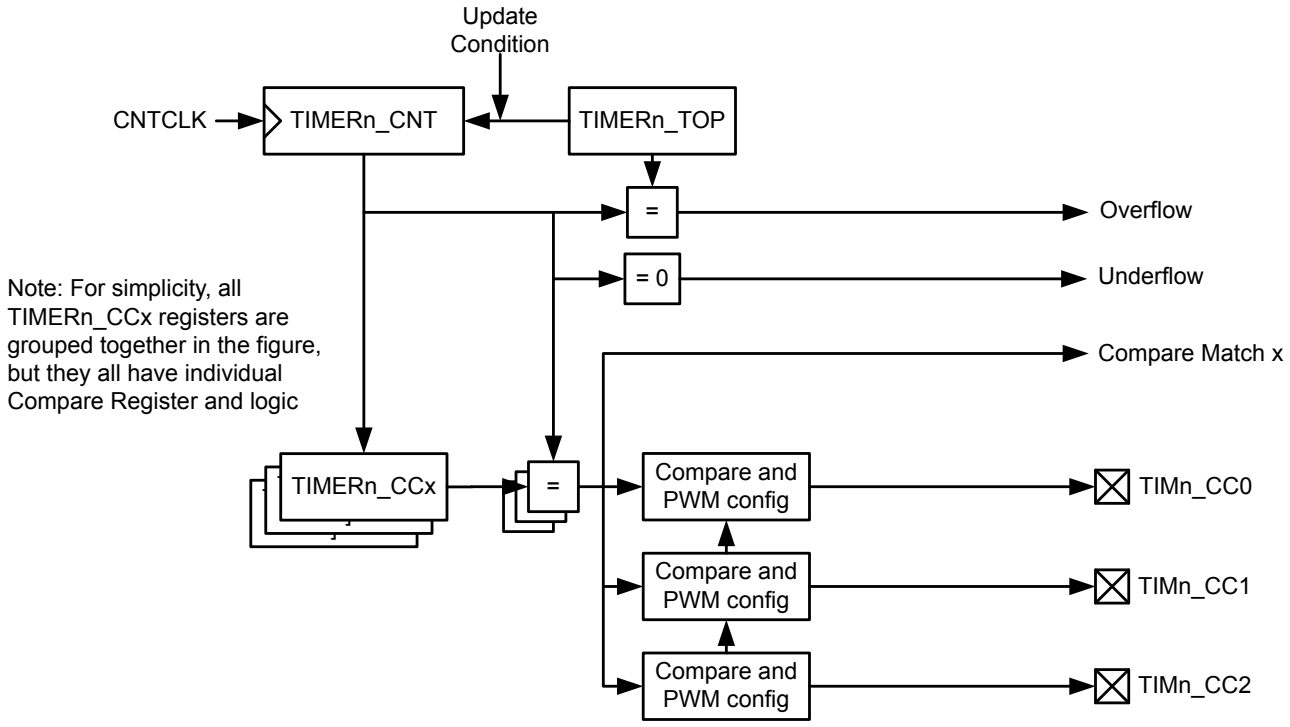


Figure 19.14. TIMER Block Diagram Showing Comparison Functionality

The compare output is delayed by one cycle to allow for full 0% to 100% PWM generation. If occurring in the same cycle, match action will have priority over overflow or underflow action.

The input selected (through PRSSEL in PRS_CONSUMER_TIMERn_CCx, INSEL and FILT in TIMERn_CCx_CFG) for the CC channel will also be sampled on compare match and the result is found in the CCPOL bits in TIMERn_STATUS. It is also possible to configure the CCPOL to always track the inputs by setting ATI in TIMERn_CFG.

Note: When using synchronous PRS sources, it is recommended to configure the PRS consumer registers prior to selecting PRS triggering to avoid any false triggers.

The COIST bit in TIMERn_CCx_CFG is the initial state of the compare/PWM output. The COIST bit can also be used as an initial value to the compare outputs on a reload-start when RSSCOIST is set in TIMERn_CFG. Also the resulting output can be inverted by setting OUTINV in TIMERn_CCx_CTRL. It is recommended to turn off the CC channel before configuring the output state to avoid any unwanted pulses on the output. The CC channel can be turned off by setting MODE to OFF in TIMER_CCx_CFG. The following figure shows the output logic for the TIMER module.

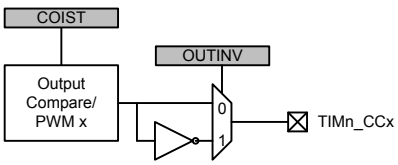


Figure 19.15. TIMER Output Logic

19.3.3.6 Compare Mode Registers

When running in output compare or PWM mode, the value in TIMERn_CCx_OC will be compared against the count value. In Compare mode the output can be configured to toggle, clear or set on compare match, overflow, and underflow through the CMOA, COFOA and CUFOA fields in TIMERn_CCx_CTRL. TIMERn_CCx_OC can be accessed directly or through the buffer register TIMERn_CCx_OCB, see Figure 19.16 TIMER Output Compare/PWM Buffer Functionality Detail on page 494. When writing to the buffer register, the value in TIMERn_CCx_OCB will be written to TIMERn_CCx_OC on the next *update event*. This functionality ensures glitch free PWM outputs. The OCBV flag in TIMERn_STATUS indicates whether the TIMERn_CCx_OCB register contains data that has not yet been written to the TIMERn_CCx_OC register. Note that when writing 0 to TIMERn_CCx_OCB in up-down count mode the OC value is updated when the timer counts from 0 to 1. Thus, the compare match for the next period will not happen until the timer reaches 0 again on the way down.

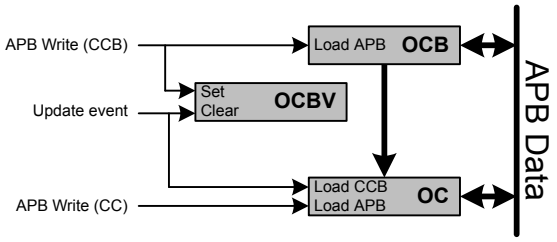


Figure 19.16. TIMER Output Compare/PWM Buffer Functionality Detail

19.3.3.7 Frequency Generation (FRG)

Frequency generation (see Figure 19.17 TIMER Up-count Frequency Generation on page 495) can be achieved in compare mode by:

- Setting the counter in up-count mode
- Enabling buffering of the TOP value.
- Setting the CC channels overflow action to toggle

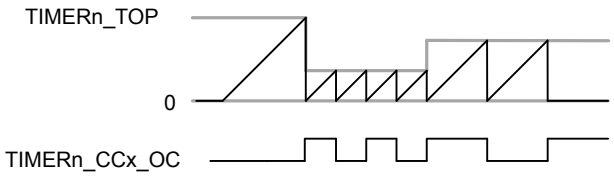


Figure 19.17. TIMER Up-count Frequency Generation

The output frequency is given by Figure 19.18 TIMER Up-count Frequency Generation Equation on page 495

$$f_{FRG} = f_{HFPERCLK\_TIMERn} / [2 \times (PRESC + 1) \times (TOP + 1)]$$

Figure 19.18. TIMER Up-count Frequency Generation Equation

The figure below provides cycle accurate timing and event generation information for frequency generation.

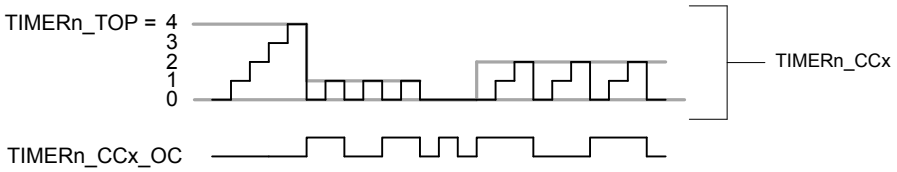


Figure 19.19. TIMER Up-count Frequency Generation Detail

19.3.3.8 Pulse-Width Modulation (PWM)

In PWM mode, TIMERn_CCx_OC is buffered to avoid glitches in the output. The settings in the Compare Output Action configuration bits are ignored in PWM mode and PWM generation is only supported for up-count and up/down-count mode.

19.3.3.9 Up-count (Single-slope) PWM

If the counter is set to up-count and the compare/capture channel is put in PWM mode, single slope PWM output will be generated (see Figure 19.20 TIMER Up-count PWM Generation on page 496). In up-count mode the PWM period is TOP+1 cycles and the PWM output will be high for a number of cycles equal to TIMERn_CCx_OC. This means that a constant high output is achieved by setting TIMERn_CCx_OC to TOP+1 or higher. The PWM resolution (in bits) is then given by Figure 19.21 TIMER Up-count PWM Resolution Equation on page 496.

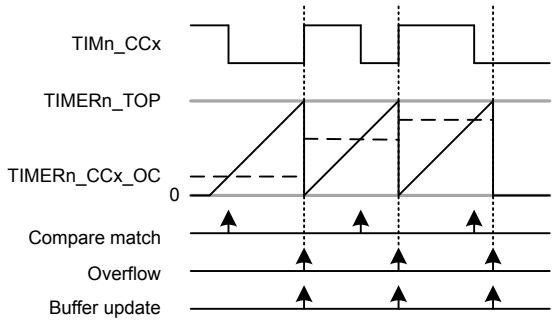


Figure 19.20. TIMER Up-count PWM Generation

$$R_{PWM_{up}} = \log(TOP+1)/\log(2)$$

Figure 19.21. TIMER Up-count PWM Resolution Equation

The PWM frequency is given by Figure 19.22 TIMER Up-count PWM Frequency Equation on page 496:

$$f_{PWM_{up}} = f_{HFPERCLK\_TIMERn} / [(PRESC + 1) \times (TOP + 1)]$$

Figure 19.22. TIMER Up-count PWM Frequency Equation

The high duty cycle is given by Figure 19.23 TIMER Up-count Duty Cycle Equation on page 496

$$DS_{up} = OCx/(TOP+1)$$

Figure 19.23. TIMER Up-count Duty Cycle Equation

The figure below provides cycle accurate timing and event generation information for up-count mode.

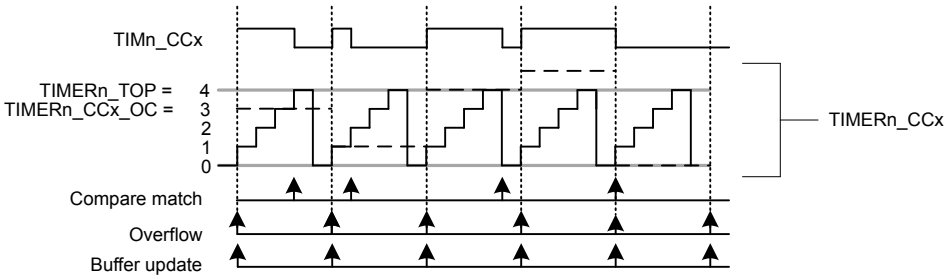


Figure 19.24. TIMER Up-count PWM Generation Detail

19.3.3.10 2x Count Mode (Up-count)

When the timer is set in 2x mode, the TIMER will count up by two for every (prescaled) clock. This will in effect make any odd Top value be rounded down to the closest even number. Similarly, any odd OC value will generate a match on the closest lower even value as shown in Figure 19.25 TIMER CC out in 2x mode on page 497

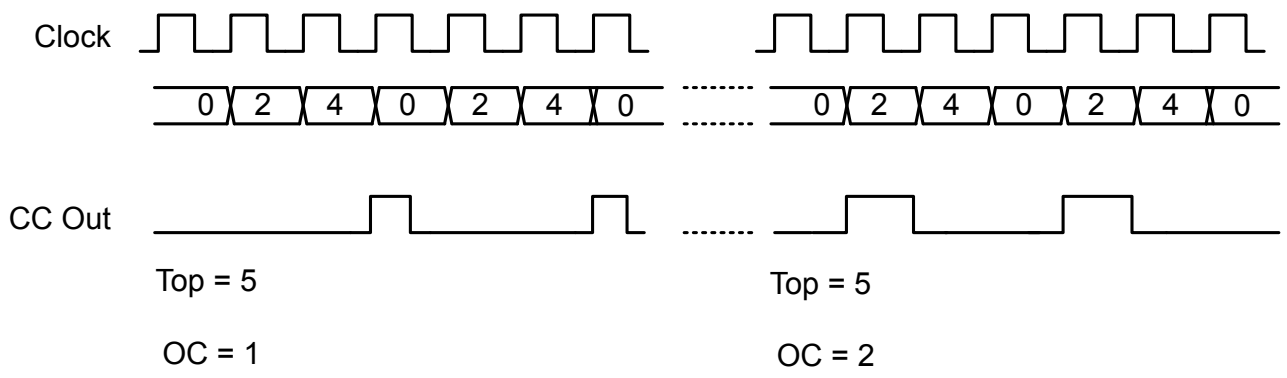


Figure 19.25. TIMER CC out in 2x mode

The PWM resolution is given by Figure 19.26 TIMER 2x PWM Resolution Equation on page 497.

$$R_{PWM_{2xmode}} = \log(TOP/2+1)/\log(2)$$

Figure 19.26. TIMER 2x PWM Resolution Equation

The PWM frequency is given by Figure 19.27 TIMER 2x Mode PWM Frequency Equation(Up-count) on page 497:

$$f_{PWM_{2xmode}} = f_{HFPERCLK\_TIMERn}/ [(PRESC + 1) \times (\text{floor}(TOP/2)+1)]$$

Figure 19.27. TIMER 2x Mode PWM Frequency Equation(Up-count)

The high duty cycle is given by Figure 19.28 TIMER 2x Mode Duty Cycle Equation on page 497

$$DS_{2xmode} = OCx/((\text{floor}(TOP/2)+1)*2)$$

Figure 19.28. TIMER 2x Mode Duty Cycle Equation

19.3.3.11 Up/Down-count (Dual-slope) PWM

If the counter is set to up-down count and the compare/capture channel is put in PWM mode, dual slope PWM output will be generated by Figure 19.29 TIMER Up/Down-count PWM Generation on page 498.The resolution (in bits) is given by Figure 19.30 TIMER Up/Down-count PWM Resolution Equation on page 498.

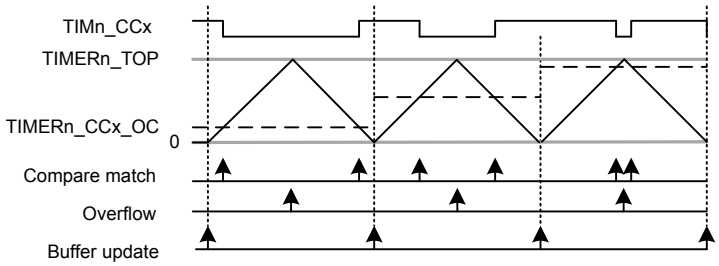


Figure 19.29. TIMER Up/Down-count PWM Generation

$$R_{PWM_{up/down}} = \log(TOP+1)/\log(2)$$

Figure 19.30. TIMER Up/Down-count PWM Resolution Equation

The PWM frequency is given by Figure 19.31 TIMER Up/Down-count PWM Frequency Equation on page 498:

$$f_{PWM_{up/down}} = f_{HFPERCLK\_TIMERn}/ (2 \times (PRESC + 1) \times TOP))$$

Figure 19.31. TIMER Up/Down-count PWM Frequency Equation

The high duty cycle is given by Figure 19.32 TIMER Up/Down-count Duty Cycle Equation on page 498

$$DS_{up/down} = OCx/TOP$$

Figure 19.32. TIMER Up/Down-count Duty Cycle Equation

The figure below provides cycle accurate timing and event generation information for up-count mode.

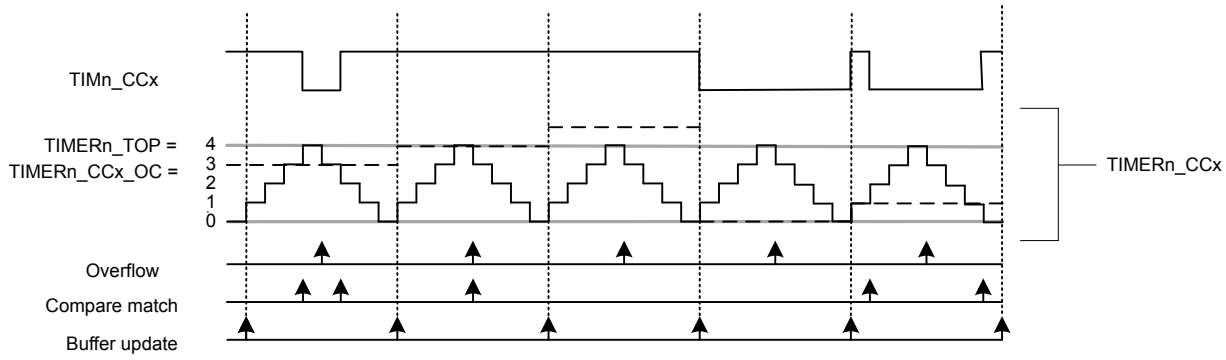


Figure 19.33. TIMER Up/Down-count PWM Generation

19.3.3.12 2x Count Mode (Up/Down-count)

When the timer is set in 2x mode, the TIMER will count up/down by two. This will in effect make any odd Top value be rounded down to the closest even number. Similarly, any odd OC value will generate a match on the closest lower even value as shown in Figure 19.34 TIMER CC out in 2x mode on page 499

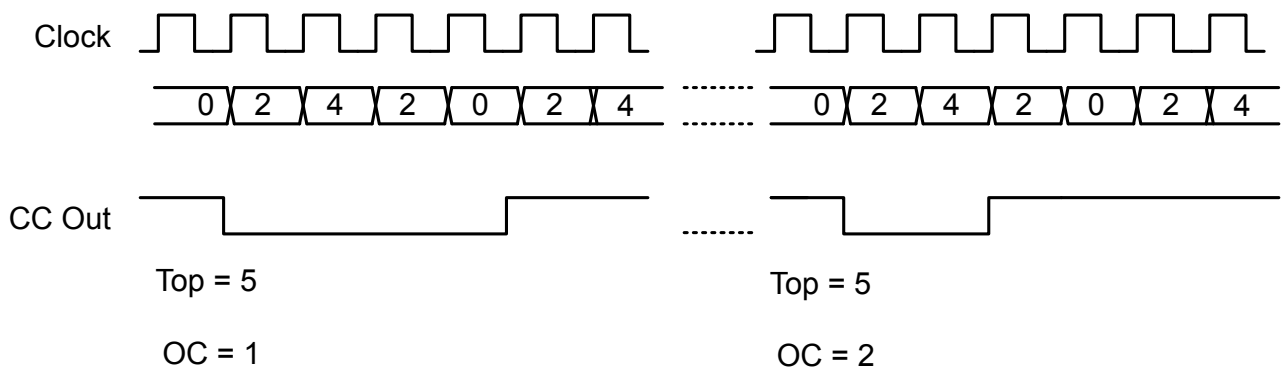


Figure 19.34. TIMER CC out in 2x mode

Figure 19.35 TIMER 2x PWM Resolution Equation on page 499.

$$R_{PWM_{2xmode}} = \log(TOP/2+1)/\log(2)$$

Figure 19.35. TIMER 2x PWM Resolution Equation

The PWM frequency is given by Figure 19.36 TIMER 2x Mode PWM Frequency Equation(Up/Down-count) on page 499:

$$f_{PWM_{2xmode}} = f_{HFPERCLK\_TIMERn}/ (2 \times (PRESC + 1) \times (\text{floor}(TOP/2)))$$

Figure 19.36. TIMER 2x Mode PWM Frequency Equation(Up/Down-count)

The high duty cycle is given by two equations based on the OCx values.Figure 19.37 TIMER 2x Mode Duty Cycle Equation for OCx = 1 or OCx = even on page 499 and Figure 19.38 TIMER 2x Mode Duty Cycle Equation for all other OCx = odd values on page 499

$$DS_{2xmode} = (OCx*2)/(\text{floor}(TOP/2)*4)$$

Figure 19.37. TIMER 2x Mode Duty Cycle Equation for OCx = 1 or OCx = even

$$DS_{2xmode} = (OCx*2 - OCx)/(\text{floor}(TOP/2)*4)$$

Figure 19.38. TIMER 2x Mode Duty Cycle Equation for all other OCx = odd values

19.3.3.13 Re-Timing PWM Outputs

PWM outputs are normally synchrous to the TIMER peripheral clock. However for radio applications, it can be desirable to synchronize PWM edges to radio clocks to reduce the interference with RF signalling.

Re-timing is enabled by setting the RETIMEEN bit in TIMERn_CFG to 1. When RETIMEEN is enabled, PWM X2CNT mode should not be enabled. Doing so may result in unpredictable PWM behavior.

Direct re-timing is supported at peripheral clock frequencies up to 50 MHz. For higher peripheral clock frequencies, set the RETIMESEL bit in TIMERn_CFG to 1. This allows PWM outputs to be re-timed at frequencies up to 80 MHz, but will introduce up to 1 $HFPERCLK_{TIMERn}$ cycle of jitter between the PWM outputs.

19.3.3.14 Timer Configuration Lock

To prevent software errors from making changes to the timer configuration, a configuration lock is available. Writing any value but 0xCE80 to LOCKKEY in TIMERn_LOCK will lock writes to TIMERn_CTRL, TIMERn_CFG, TIMERn_CMD, TIMERn_TOP, TIMERn_TOPB, TIMERn_CNT, TIMERn_CCx_CTRL, TIMERn_CCx_CFG, TIMERn_CCx_OC, and TIMERn_CCx_OCB. To unlock the registers, write 0xCE80 to LOCKKEY in TIMERn_LOCK. The value of TIMERLOCKSTATUS in TIMERn_STATUS is 1 when the lock is active, and 0 when the registers are unlocked.

19.3.4 Dead-Time Insertion Unit

Some timer modules include a Dead-Time Insertion unit suitable for motor control applications. Refer to the device data sheet to check which timer instances have this feature. The example settings in this section are for TIMER0, but identical settings can be used for other timers with DTI as well. The Dead-Time Insertion Unit aims to make control of brushless DC (BLDC) motors safer and more efficient by introducing complementary PWM outputs with dead-time insertion and fault handling, see Figure 19.39 TIMER Dead-Time Insertion Unit Overview on page 501.

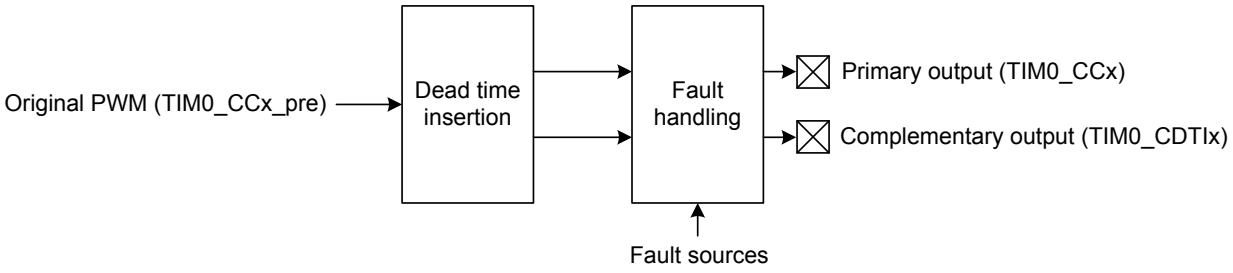


Figure 19.39. TIMER Dead-Time Insertion Unit Overview

When used for motor control, the PWM outputs TIM0_CC0, TIM0_CC1 and TIM0_CC2 are often connected to the high-side transistors of a triple half-bridge setup (UH, VH and WH), and the complementary outputs connected to the respective low-side transistors (UL, VL, WL shown in Figure 19.40 TIMER Triple Half-Bridge on page 501). Transistors used in such a bridge often do not open/close instantaneously, and using the exact complementary inputs for the high and low side of a half-bridge may result in situations where both gates are open. This can give unnecessary current-draw and short circuit the power supply. The DTI unit provides dead-time insertion to deal with this problem.

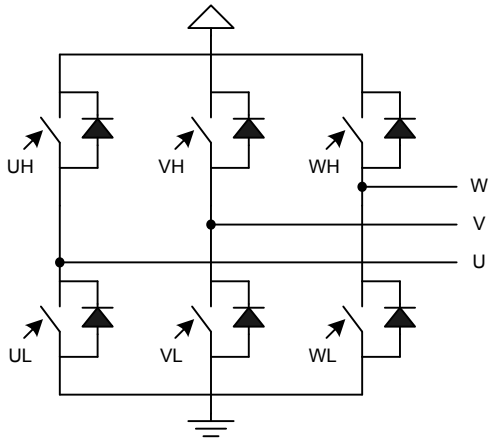


Figure 19.40. TIMER Triple Half-Bridge

For each of the 3 compare-match outputs of TIMER0, an additional complementary output is provided by the DTI unit. These outputs, named TIM0_CDTI0, TIM0_CDTI1 and TIM0_CDTI2 are provided to make control of e.g. 3-channel BLDC or permanent magnet AC (PMAC) motors possible using only a single timer, see Figure 19.41 TIMER Overview of Dead-Time Insertion Block for a Single PWM channel on page 502.

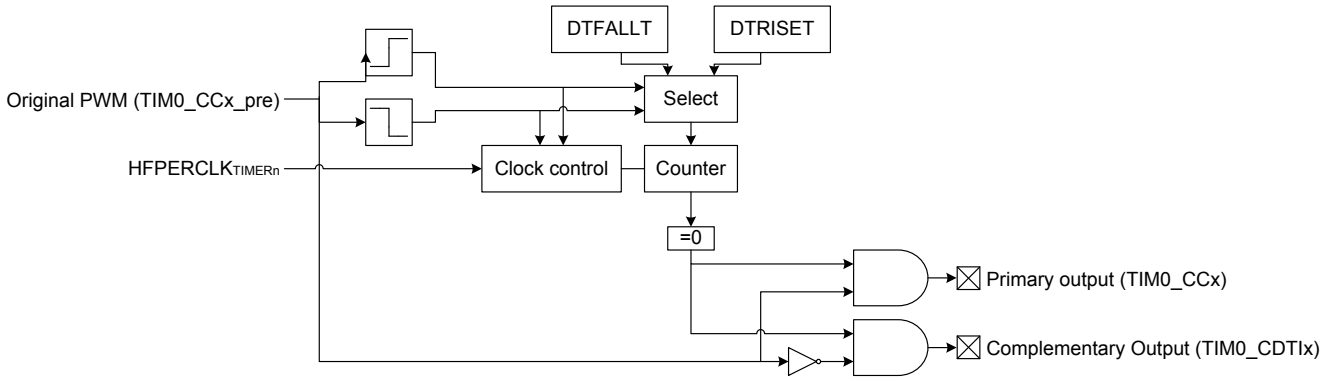


Figure 19.41. TIMER Overview of Dead-Time Insertion Block for a Single PWM channel

The DTI unit is enabled by setting DTEN in TIMER0_DTCFG. In addition to providing the complementary outputs, the DTI unit then also overrides the compare match outputs from the timer.

The DTI unit gives the rising edges of the PWM outputs and the rising edges of the complementary PWM outputs a configurable time delay. By doing this, the DTI unit introduces a dead-time where both the primary and complementary outputs in a pair are inactive as seen in Figure 19.42 TIMER Polarity of Both Signals are Set as Active-High on page 502.

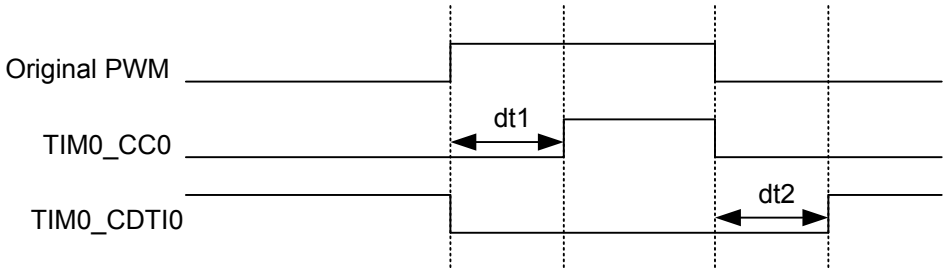


Figure 19.42. TIMER Polarity of Both Signals are Set as Active-High

Dead-time is specified individually for the rising and falling edge of the original PWM. These values are shared across all the three PWM channels of the DTI unit. A single prescaler value is provided for the DTI unit, meaning that both the rising and falling edge dead-times share prescaler value. The prescaler divides the $HFPERCLK_{TIMER0}$ by a configurable factor between 1 and 1024, which is set in the DTPRESC field in TIMER0_DTTIMECFG. The rising and falling edge dead-times are configured in DTRISET and DTFALLT in TIMER0_DTTIMECFG to any number between 1-64 $HFPERCLK_{TIMER0}$ cycles.

The DTAR and DTFATS bits in TIMER0_DTCFG control the DTI output behavior when the timer stops. By default the DTI block stops when the timer is stopped. Setting the DTAR bit will cause the DTI output on channel 0 to continue when the timer is stopped. DTAR effects only channel 0. See 19.3.4.2 PRS Channel as a Source for an example of when this can be used. While in this mode the undivided $HFPERCLK_{TIMER0}$ (DTPRESC=0) is always used regardless of the programmed DTPRESC value in TIMER0_DTTIMECFG. This means that rise and fall dead times are calculated assuming DTPRESC = 0.

When the timer stops, DTI outputs are frozen by default, preserving their last state. To allow the outputs to go to a safe state, program the DTFA field of the TIMER0_DTFCFG register to the safe values and set the DTFATS bitfield in the TIMER0_DTCFG register. Note that when DTAR is also set, DTAR has priority over DTFATS for DTI channel 0 output.

The following table shows the DTI output when the timer is halted.

Table 19.3. DTI Output When Timer Halted

DTAR	DTFATS	State
0	0	frozen
0	1	safe
1	0	running
1	1	running

19.3.4.1 Output Polarity

The value of the primary and complementary outputs in a pair will never be set active at the same time by the DTI unit. The polarity of the outputs can be changed if this is required by the application. The active values of the primary and complementary outputs are set by the DTIPOL and DTCINV bits in the TIMER0_DTCTRL register. The DTIPOL bit of this register specifies the base polarity. If DTIPOL = 0, then the outputs are active-high, and if DTIPOL = 1 they are active-low. The relative phase of the primary and complementary outputs is not changed by DTIPOL, as the polarity of both outputs is changed, see Figure 19.43 TIMER Output Polarities on page 503.

In some applications, it may be required that the primary outputs are active-high, while the complementary outputs are active-low. This can be accomplished by manipulating the DTCINV bit of the TIMER0_DTCTRL register, which inverts the polarity of the complementary outputs relative to the primary outputs. As an example, DTIPOL = 0 and DTCINV = 0 results in outputs with opposite phase and active-high states. Similarly, DTIPOL = 1 and DTCINV = 1 results in outputs with equal phase and the primary output will be active-high while the complementary will be active-low.

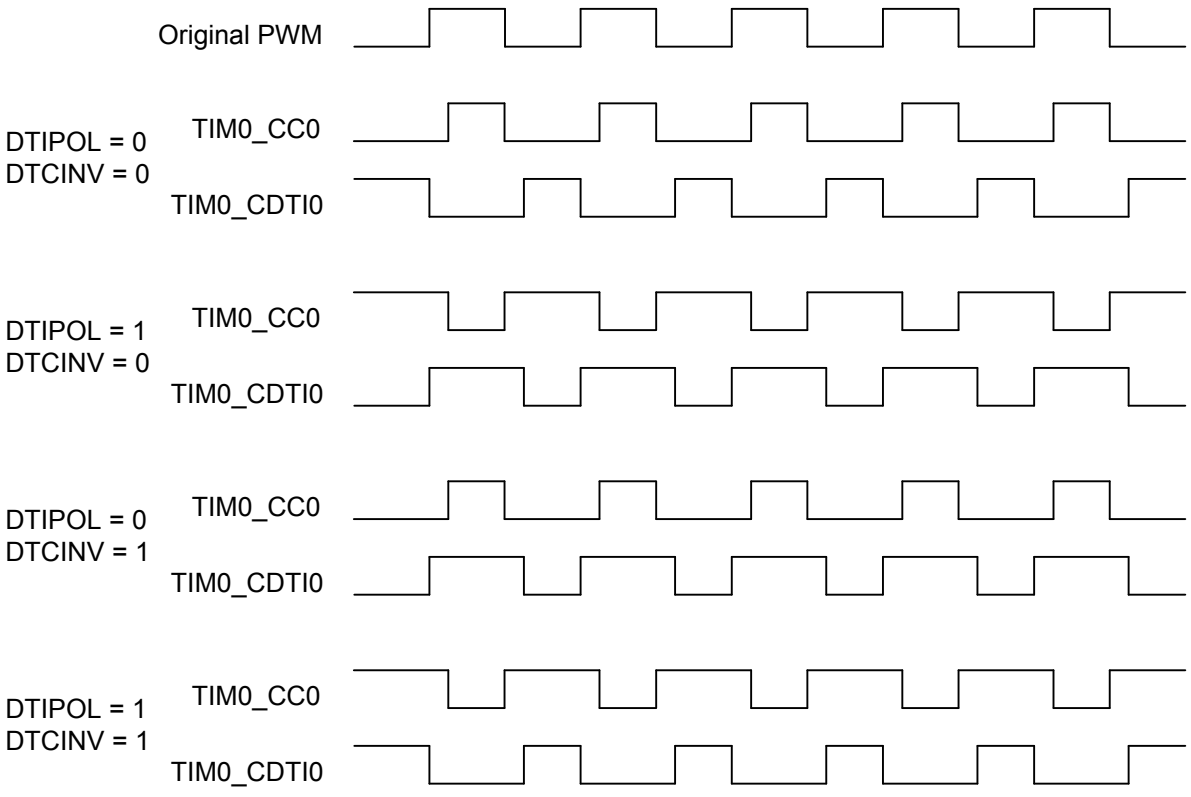


Figure 19.43. TIMER Output Polarities

Output generation on the individual DTI outputs can be disabled by configuring TIMER0_DTOGEN. When output generation on an output is disabled that output will go to and stay in its inactive state.

19.3.4.2 PRS Channel as a Source

A PRS channel can be used as input to the DTI module instead of the PWM output from the timer for DTI channel 0. Setting DTPRSEN in TIMER0_DTCFG will override the source of the first DTI channel, driving TIM0_CC0 and TIM0_CDTI0, with the value on the PRS channel. The rest of the DTI channels will continue to be driven by the PWM output from the timer. The input PRS channel is chosen within the PRS module with PRSSEL in the PRS_CONSUMER_TIMERn_DTI register. Note that the timer must be running even when PRS is used as the DTI source. However, if it is required to keep the DTI channel 0 running even when the timer is stopped, set DTAR in TIMER0_DTCFG. When this bit is set, it uses DTPRESC=0 regardless of the value programmed in DTPRESC in TIMER0_DTTIMECFG.

Note: When using synchronous PRS sources, it is recommended to configure the PRS consumer registers prior to selecting PRS triggering to avoid any false triggers.

The DTI prescaler, set by DTPRESC in TIMER0_DTTIMECFG determines the accuracy with which the DTI can insert dead-time into a PRS signal. The maximum dead-time error equals DTIPRESC+1 clock cycles. With DTIPRESC = 0, the inserted dead-times are therefore accurate, but they may be inaccurate for larger prescaler settings.

19.3.4.3 Fault Handling

The fault handling system of the DTI unit allows the outputs of the DTI unit to be put in a well-defined state in case of a fault. This hardware fault handling system enables a fast reaction to faults, reducing the possibility of damage to the system.

The fault sources which trigger a fault in the DTI module are determined by the bitfields of TIMER0_DTFCFG register. Any combination of the available error sources can be selected:

- PRS source 1, determined by PRSSEL in PRS_CONSUMER_TIMERn_DTIFS1
- PRS source 2, determined by PRSSEL in PRS_CONSUMER_TIMERn_DTIFS2
- Debugger
- Core Lockup
- EM2 or EM3 Entry

One or two PRS channels can be used as an error source. When PRS source 1 is selected as an error source, PRSSEL in PRS_CONSUMER_TIMERn_DTIFS1 determines which PRS channel is used for this source. PRSSEL in PRS_CONSUMER_TIMERn_DTIFS2 determines which PRS channel is selected as PRS source 2. Note that for Core Lockup, the LOCKUPRDIS in RMU_CTRL must be set. Otherwise this will generate a full reset of the chip.

Note: When using synchronous PRS sources, it is recommended to configure the PRS consumer registers prior to selecting PRS triggering to avoid any false triggers.

19.3.4.4 Action on Fault

When a fault occurs, the bit representing the fault source is set in TIMER0_DTFAULT register, and the outputs from the DTI unit are set to a well-defined state. The following options are available, and can be enabled by configuring DTFA in TIMER0_DTFCFG:

- Set outputs to inactive level
- Clear outputs
- Tristate outputs

With the first option enabled, the output state in case of a fault depends on the polarity settings for the individual outputs. An output set to be active high will be set low if a fault is detected, while an output set to be active low will be driven high.

When a fault occurs, the fault source(s) can be read out from TIMER0_DTFAULT register.

Additionally a fault action can also be triggered when the timer stops if DTFATS in TIMER0_DTCFG is set. This allows the DTI output to go to safe state specified by DTFA in TIMER0_DTFCFG when the timer stops. When DTAR and DTFATS in TIMER0_DTCFG are both set, DTI channel 0 keeps running even when the timer stops. This is useful when DTI channel 0 has an input coming from PRS.

19.3.4.5 Exiting Fault State

When a fault is triggered by the PRS system, software intervention is required to re-enable the outputs of the DTI unit. This is done by manually clearing bits in the TIMER0_DTFAULT register. If the fault source as determined by checking TIMER0_DTFAULT is the debugger alone, the outputs can be automatically restarted when the debugger exits. To enable automatic restart set DTDAS in TIMER0_DCTFG. When an automatic restart occurs the DTDBGF bit in TIMER0_DTFAULT will be automatically cleared by hardware. If any other bits in the TIMER0_DTFAULT register are set when the hardware clears DTDBGF the DTI module will not exit the fault state.

19.3.4.6 DTI Configuration Lock

To prevent software errors from making changes to the DTI configuration, a configuration lock is available. Writing any value but 0xCE80 to LOCKKEY in TIMER0_DTLOCK locks writes to registers TIMER0_DTCFG, TIMER0_DTFCFG, TIMER0_DTCTRL, and TIMER0_DTTIMECFG. To unlock the registers, write 0xCE80 to LOCKKEY in TIMER0_DTLOCK. The value of DTILOCKSTATUS in TIMERn_STATUS is 1 when the lock is active, and 0 when the registers are unlocked.

19.3.5 Debug Mode

When the CPU is halted in debug mode, the timer can be configured to either continue to run or to be frozen. This is configured in DEBUGRUN in TIMERn_CFG.

19.3.6 Interrupts, DMA and PRS Output

The timer can generate several type of output events:

- Counter Underflow
- Counter Overflow
- Quadrature Decoder Direction Change
- Compare match or input capture (one per compare/capture channel)

Each of the events has its own interrupt flag. Also, there are interrupt flags for each compare/capture channel which are set on FIFO overflow or underflow in capture mode. FIFO overflow happens when a new capture over-writes an old unread capture in TIMERn_CCx_ICF. FIFO underflow happens when software reads TIMERn_CCx_ICF while the FIFO is empty.

If the interrupt flags are set and the corresponding interrupt enable bits in TIMERn_IEN are set high, the timer will send out an interrupt request. Each of the events may optionally trigger signals to PRS channels. The PRSCONF field in TIMERn_CCx_CFG determines how PRS events are generated. When PRSCONF is set to PULSE, and event will lead to a one $HFPERCLK_{TIMERn}$ cycle high pulse on individual PRS outputs. Setting PRSCONF to LEVEL will make the PRS output follow the compare match output. Interrupts are cleared by setting the corresponding bit in the TIMERn_IFC register.

Each of the events will also set a DMA request when they occur. The different DMA requests are cleared when certain acknowledge conditions are met, see Table 19.4 TIMER DMA Events on page 505. Events which clear the DMA requests do not clear interrupt flags. Software must still manually clear the interrupt flag if interrupts are in use.

If DMACLRACT is set in TIMERn_CFG, the DMA request is cleared when the triggered DMA channel is active, without having to access any timer registers. This is useful in cases where a timer event is used to trigger a DMA transfer in output compare or PWM mode that does not target the OC or OCB registers. DMACLRACT is not applicable in input capture mode.

Table 19.4. TIMER DMA Events

Event	Acknowledge/Clear
Underflow/Overflow	Read or write to TIMERn_CNT or TIMERn_TOPB
CC0 Input Capture - ICFWLFULL0 flag set	ICFEMPTY0 flag set (read FIFO via TIMERn_CC0_ICF)
CC1 Input Capture - ICFWLFULL1 flag set	ICFEMPTY1 flag set (read FIFO via TIMERn_CC1_ICF)
CC2 Input Capture - ICFWLFULL2 flag set	ICFEMPTY2 flag set (read FIFO via TIMERn_CC2_ICF)
CC3 Input Capture - ICFWLFULL3 flag set	ICFEMPTY3 flag set (read FIFO via TIMERn_CC3_ICF)
CC0 Output Compare / PWM - Match event	Write TIMERn_CC0_OC or TIMERn_CC0_OCB
CC1 Output Compare / PWM - Match event	Write TIMERn_CC1_OC or TIMERn_CC1_OCB
CC2 Output Compare / PWM - Match event	Write TIMERn_CC2_OC or TIMERn_CC2_OCB
CC3 Output Compare / PWM - Match event	Write TIMERn_CC3_OC or TIMERn_CC3_OCB

19.3.7 GPIO Input/Output

The TIMn_CCx inputs/outputs and TIMn_CDTIx outputs are accessible as alternate functions through GPIO. Each pin connection can be enabled/disabled separately using the GPIO module control registers. See the device data sheet for the available locations for each signal.

19.4 TIMER Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	TIMER_IPVERSION	R	IP version ID
0x004	TIMER_CFG	RW CONFIG	Configuration Register
0x008	TIMER_CTRL	RW SYNC	Control Register
0x00C	TIMER_CMD	W SYNC	Command Register
0x010	TIMER_STATUS	RH	Status Register
0x014	TIMER_IF	RWH INTFLAG	Interrupt Flag Register
0x018	TIMER_IEN	RW	Interrupt Enable Register
0x01C	TIMER_TOP	RWH SYNC	Counter Top Value Register
0x020	TIMER_TOPB	RW SYNC	Counter Top Value Buffer Register
0x024	TIMER_CNT	RWH SYNC	Counter Value Register
0x02C	TIMER_LOCK	W	TIMER Configuration Lock Register
0x030	TIMER_EN	RW ENABLE	module en
0x060	TIMER_CCx_CFG	RW CONFIG	CC Channel Configuration Register
0x064	TIMER_CCx_CTRL	RW SYNC	CC Channel Control Register
0x068	TIMER_CCx_OC	RWH SYNC	OC Channel Value Register
0x070	TIMER_CCx_OCB	RW SYNC	OC Channel Value Buffer Register
0x074	TIMER_CCx_ICF	R(r)H	IC Channel Value Register
0x078	TIMER_CCx_ICOF	RH SYNC	IC Channel Value Overflow Register
0x0E0	TIMER_DTCFG	RW CONFIG	DTI Configuration Register
0x0E4	TIMER_DTTIMECFG	RW CONFIG	DTI Time Configuration Register
0x0E8	TIMER_DTFCFG	RW CONFIG	DTI Fault Configuration Register
0x0EC	TIMER_DTCTRL	RW SYNC	DTI Control Register
0x0F0	TIMER_DTOGEN	RW SYNC	DTI Output Generation Enable Register
0x0F4	TIMER_DTFAULT	RH	DTI Fault Register
0x0F8	TIMER_DTFAULTC	W SYNC	DTI Fault Clear Register
0x0FC	TIMER_DTLOCK	W	DTI Configuration Lock Register
0x1000	TIMER_IPVERSION_SET	R	IP version ID
0x1004	TIMER_CFG_SET	RW CONFIG	Configuration Register
0x1008	TIMER_CTRL_SET	RW SYNC	Control Register
0x100C	TIMER_CMD_SET	W SYNC	Command Register
0x1010	TIMER_STATUS_SET	RH	Status Register
0x1014	TIMER_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1018	TIMER_IEN_SET	RW	Interrupt Enable Register
0x101C	TIMER_TOP_SET	RWH SYNC	Counter Top Value Register
0x1020	TIMER_TOPB_SET	RW SYNC	Counter Top Value Buffer Register

Offset	Name	Type	Description
0x1024	TIMER_CNT_SET	RWH SYNC	Counter Value Register
0x102C	TIMER_LOCK_SET	W	TIMER Configuration Lock Register
0x1030	TIMER_EN_SET	RW ENABLE	module en
0x1060	TIMER_CCx_CFG_SET	RW CONFIG	CC Channel Configuration Register
0x1064	TIMER_CCx_CTRL_SET	RW SYNC	CC Channel Control Register
0x1068	TIMER_CCx_OC_SET	RWH SYNC	OC Channel Value Register
0x1070	TIMER_CCx_OCB_SET	RW SYNC	OC Channel Value Buffer Register
0x1074	TIMER_CCx_ICF_SET	R(r)H	IC Channel Value Register
0x1078	TIMER_CCx_ICOF_SET	RH SYNC	IC Channel Value Overflow Register
0x10E0	TIMER_DTCFG_SET	RW CONFIG	DTI Configuration Register
0x10E4	TIMER_DTTIMECFG_SET	RW CONFIG	DTI Time Configuration Register
0x10E8	TIMER_DTFCFG_SET	RW CONFIG	DTI Fault Configuration Register
0x10EC	TIMER_DTCTRL_SET	RW SYNC	DTI Control Register
0x10F0	TIMER_DTOGEN_SET	RW SYNC	DTI Output Generation Enable Register
0x10F4	TIMER_DTFAULT_SET	RH	DTI Fault Register
0x10F8	TIMER_DTFAULTC_SET	W SYNC	DTI Fault Clear Register
0x10FC	TIMER_DTLOCK_SET	W	DTI Configuration Lock Register
0x2000	TIMER_IPVERSION_CLR	R	IP version ID
0x2004	TIMER_CFG_CLR	RW CONFIG	Configuration Register
0x2008	TIMER_CTRL_CLR	RW SYNC	Control Register
0x200C	TIMER_CMD_CLR	W SYNC	Command Register
0x2010	TIMER_STATUS_CLR	RH	Status Register
0x2014	TIMER_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2018	TIMER_IEN_CLR	RW	Interrupt Enable Register
0x201C	TIMER_TOP_CLR	RWH SYNC	Counter Top Value Register
0x2020	TIMER_TOPB_CLR	RW SYNC	Counter Top Value Buffer Register
0x2024	TIMER_CNT_CLR	RWH SYNC	Counter Value Register
0x202C	TIMER_LOCK_CLR	W	TIMER Configuration Lock Register
0x2030	TIMER_EN_CLR	RW ENABLE	module en
0x2060	TIMER_CCx_CFG_CLR	RW CONFIG	CC Channel Configuration Register
0x2064	TIMER_CCx_CTRL_CLR	RW SYNC	CC Channel Control Register
0x2068	TIMER_CCx_OC_CLR	RWH SYNC	OC Channel Value Register
0x2070	TIMER_CCx_OCB_CLR	RW SYNC	OC Channel Value Buffer Register
0x2074	TIMER_CCx_ICF_CLR	R(r)H	IC Channel Value Register
0x2078	TIMER_CCx_ICOF_CLR	RH SYNC	IC Channel Value Overflow Register
0x20E0	TIMER_DTCFG_CLR	RW CONFIG	DTI Configuration Register
0x20E4	TIMER_DTTIMECFG_CLR	RW CONFIG	DTI Time Configuration Register

Offset	Name	Type	Description
0x20E8	TIMER_DTFCFG_CLR	RW CONFIG	DTI Fault Configuration Register
0x20EC	TIMER_DTCTRL_CLR	RW SYNC	DTI Control Register
0x20F0	TIMER_DTOGEN_CLR	RW SYNC	DTI Output Generation Enable Register
0x20F4	TIMER_DTFAULT_CLR	RH	DTI Fault Register
0x20F8	TIMER_DTFAULTC_CLR	W SYNC	DTI Fault Clear Register
0x20FC	TIMER_DTLOCK_CLR	W	DTI Configuration Lock Register
0x3000	TIMER_IPVERSION_TGL	R	IP version ID
0x3004	TIMER_CFG_TGL	RW CONFIG	Configuration Register
0x3008	TIMER_CTRL_TGL	RW SYNC	Control Register
0x300C	TIMER_CMD_TGL	W SYNC	Command Register
0x3010	TIMER_STATUS_TGL	RH	Status Register
0x3014	TIMER_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3018	TIMER_IEN_TGL	RW	Interrupt Enable Register
0x301C	TIMER_TOP_TGL	RWH SYNC	Counter Top Value Register
0x3020	TIMER_TOPB_TGL	RW SYNC	Counter Top Value Buffer Register
0x3024	TIMER_CNT_TGL	RWH SYNC	Counter Value Register
0x302C	TIMER_LOCK_TGL	W	TIMER Configuration Lock Register
0x3030	TIMER_EN_TGL	RW ENABLE	module en
0x3060	TIMER_CCx_CFG_TGL	RW CONFIG	CC Channel Configuration Register
0x3064	TIMER_CCx_CTRL_TGL	RW SYNC	CC Channel Control Register
0x3068	TIMER_CCx_OC_TGL	RWH SYNC	OC Channel Value Register
0x3070	TIMER_CCx_OCB_TGL	RW SYNC	OC Channel Value Buffer Register
0x3074	TIMER_CCx_ICF_TGL	R(r)H	IC Channel Value Register
0x3078	TIMER_CCx_ICOF_TGL	RH SYNC	IC Channel Value Overflow Register
0x30E0	TIMER_DTCFG_TGL	RW CONFIG	DTI Configuration Register
0x30E4	TIMER_DTTIMECFG_TGL	RW CONFIG	DTI Time Configuration Register
0x30E8	TIMER_DTFCFG_TGL	RW CONFIG	DTI Fault Configuration Register
0x30EC	TIMER_DTCTRL_TGL	RW SYNC	DTI Control Register
0x30F0	TIMER_DTOGEN_TGL	RW SYNC	DTI Output Generation Enable Register
0x30F4	TIMER_DTFAULT_TGL	RH	DTI Fault Register
0x30F8	TIMER_DTFAULTC_TGL	W SYNC	DTI Fault Clear Register
0x30FC	TIMER_DTLOCK_TGL	W	DTI Configuration Lock Register

19.5 TIMER Register Description

19.5.1 TIMER_IPVERSION - IP version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP Version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

19.5.2 TIMER_CFG - Configuration Register

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0										0x0	0x0				0x0	0x0	0x0	0x0		0x0	0x0	0x0	0x0	0x0		0x0	
Access					RW										RW	RW				RW	RW	RW	RW		RW	RW	RW	RW	RW		RW	
Name					PRESC										RSSCOIST	ATI				RETIMESEL	DISSYNCOUT	RETIMEEN	CLKSEL		DMACLRACT	DEBUGRUN	QDM	OSMEN	SYNC		MODE	

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:18	PRESC	0x0	RW	**Prescaler Setting**

These bits select the prescaling factor for the counter clock. The selected timer clock will be divided by PRESC+1 before clocking the counter. The following modes are provided for easier software porting from Series 0 or Series 1 devices. However, the prescaler is not limited to these options.

Value	Mode	Description
0	DIV1	No prescaling
1	DIV2	Prescale by 2
3	DIV4	Prescale by 4
7	DIV8	Prescale by 8
15	DIV16	Prescale by 16
31	DIV32	Prescale by 32
63	DIV64	Prescale by 64
127	DIV128	Prescale by 128
255	DIV256	Prescale by 256
511	DIV512	Prescale by 512
1023	DIV1024	Prescale by 1024

17	RSSCOIST	0x0	RW	**Reload-Start Sets COIST**

When enabled, compare output is set to COIST value on a Reload-Start event.

16	ATI	0x0	RW	**Always Track Inputs**

Enabling ATI makes CCPOL always track the polarity of the inputs.

15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	RETIMESEL	0x0	RW	**PWM output retime select**

When RETIMEEN is set, the PWM output stage will be re-timed to synchronize edges with radio clocks and reduce RF interference. This will introduce up to 1 cycle of clock jutter between PWM outputs.

11	DISSYNCOUT	0x0	RW	**Disable Timer Start/Stop/Reload output**

When this bit is set, the Timer does not start/stop/reload other timers with SYNC bit set.

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	EN		Timer can start/stop/reload other timers with SYNC bit set
	1	DIS		Timer cannot start/stop/reload other timers with SYNC bit set
10	RETIMEEN	0x0	RW	**PWM output retimed enable**
	Enable retiming of PWM output.			
	Value	Mode		Description
	0	DISABLE		PWM outputs are not re-timed.
	1	ENABLE		PWM outputs are re-timed.
9:8	CLKSEL	0x0	RW	**Clock Source Select**
	These bits select the clock source for the timer.			
	Value	Mode		Description
	0	PRESCEM01GRPACLK		Prescaled EM01GRPACLK
	1	CC1		Compare/Capture Channel 1 Input
	2	TIMEROUF		Timer is clocked by underflow(down-count) or overflow(up-count) in the lower numbered neighbor Timer
7	DMACLRACT	0x0	RW	**DMA Request Clear on Active**
	When this bit is set, the DMA requests are cleared when the corresponding DMA channel is active. This enables the timer DMA requests to be cleared without accessing the timer.			
6	DEBUGRUN	0x0	RW	**Debug Mode Run Enable**
	Set this bit to enable timer to run in debug mode.			
	Value	Mode		Description
	0	HALT		Timer is halted in debug mode
	1	RUN		Timer is running in debug mode
5	QDM	0x0	RW	**Quadrature Decoder Mode Selection**
	This bit sets the mode for the quadrature decoder.			
	Value	Mode		Description
	0	X2		X2 mode selected
	1	X4		X4 mode selected
4	OSMEN	0x0	RW	**One-shot Mode Enable**
	Enable/disable one shot mode.			
3	SYNC	0x0	RW	**Timer Start/Stop/Reload Synchronization**
	When this bit is set, the Timer is started/stopped/reloaded by start/stop/reload commands in the other timers.			
	Value	Mode		Description
	0	DISABLE		Timer operation is unaffected by other timers.

Bit	Name	Reset	Access	Description
	1	ENABLE		Timer may be started, stopped and re-loaded from other timer instances.
2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	MODE	0x0	RW	**Timer Mode** These bits set the counting mode for the Timer. Note, when Quadrature Decoder Mode is selected (MODE = 'b11), the CLKSEL is don't care. The Timer is clocked by the Decoder Mode clock output.

Value	Mode	Description
0	UP	Up-count mode
1	DOWN	Down-count mode
2	UPDOWN	Up/down-count mode
3	QDEC	Quadrature decoder mode

19.5.3 TIMER_CTRL - Control Register

Offset	Bit Position																															
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0		0x0	
Access																												RW	RW		RW	
Name																												X2CNT	FALLA		RISEA	

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	X2CNT	0x0	RW	**2x Count Mode**
	Enable 2x count mode			
3:2	FALLA	0x0	RW	**Timer Falling Input Edge Action**
	These bits select the action taken in the counter when a falling edge occurs on the input.			

Value	Mode	Description
0	NONE	No action
1	START	Start counter without reload
2	STOP	Stop counter without reload
3	RELOADSTART	Reload and start counter

Bit	Name	Reset	Access	Description
1:0	RISEA	0x0	RW	**Timer Rising Input Edge Action**
	These bits select the action taken in the counter when a rising edge occurs on the input.			

Value	Mode	Description
0	NONE	No action
1	START	Start counter without reload
2	STOP	Stop counter without reload
3	RELOADSTART	Reload and start counter

19.5.4 TIMER_CMD - Command Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															W	W
Name																															STOP	START

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	STOP	0x0	W	**Stop Timer**
	Write a 1 to this bit to stop timer			
0	START	0x0	W	**Start Timer**
	Write a 1 to this bit to start timer			

19.5.5 TIMER_STATUS - Status Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset						0x0	0x0	0x0						0x0	0x0	0x0						0x0	0x0	0x0		0x0	0x0	0x0		0x0	0x0	0x0
Access						R	R	R						R	R	R						R	R	R		R	R	R		R	R	R
Name						CCPOL2	CCPOL1	CCPOL0						ICFEMPTY2	ICFEMPTY1	ICFEMPTY0						OCBV2	OCBV1	OCBV0		SYNCBUSY	DTILOCKSTATUS	TIMERLOCKSTATUS		TOPBV	DIR	RUNNING

Bit	Name	Reset	Access	Description
31:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	CCPOL2	0x0	R	**CCn Polarity**
	In Input Capture mode, this bit indicates the polarity of the edge that triggered capture in TIMERn_CC0_CCV. In Compare/PWM mode, this bit indicates the polarity of the selected input to CC channel 0. These bits are cleared when CCMODE is written to 0b00 (Off).			
	Value	Mode		Description
	0	LOWRISE		CC0 polarity low level/rising edge
	1	HIGHFALL		CC0 polarity high level/falling edge
25	CCPOL1	0x0	R	**CCn Polarity**
	In Input Capture mode, this bit indicates the polarity of the edge that triggered capture in TIMERn_CC0_CCV. In Compare/PWM mode, this bit indicates the polarity of the selected input to CC channel 0. These bits are cleared when CCMODE is written to 0b00 (Off).			
	Value	Mode		Description
	0	LOWRISE		CC0 polarity low level/rising edge
	1	HIGHFALL		CC0 polarity high level/falling edge
24	CCPOL0	0x0	R	**CCn Polarity**
	In Input Capture mode, this bit indicates the polarity of the edge that triggered capture in TIMERn_CC0_CCV. In Compare/PWM mode, this bit indicates the polarity of the selected input to CC channel 0. These bits are cleared when CCMODE is written to 0b00 (Off).			
	Value	Mode		Description
	0	LOWRISE		CC0 polarity low level/rising edge
	1	HIGHFALL		CC0 polarity high level/falling edge
23:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	ICFEMPTY2	0x0	R	**Input capture fifo empty**
	Set when input capture FIFO is empty			

Bit	Name	Reset	Access	Description
17	ICFEMPTY1	0x0	R	**Input capture fifo empty**
	Set when input capture FIFO is empty			
16	ICFEMPTY0	0x0	R	**Input capture fifo empty**
	Set when input capture FIFO is empty			
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	OCBV2	0x0	R	**Output Compare Buffer Valid**
	This field indicates that the TIMERn_CCx_CCVB registers contain data which have not been written to TIMERn_CCx_CCV. These bits are only used in OUTPUTCOMPARE or PWM mode and are cleared when CCMODE is written to 0b00 (Off).			
9	OCBV1	0x0	R	**Output Compare Buffer Valid**
	This field indicates that the TIMERn_CCx_CCVB registers contain data which have not been written to TIMERn_CCx_CCV. These bits are only used in OUTPUTCOMPARE or PWM mode and are cleared when CCMODE is written to 0b00 (Off).			
8	OCBV0	0x0	R	**Output Compare Buffer Valid**
	This field indicates that the TIMERn_CCx_CCVB registers contain data which have not been written to TIMERn_CCx_CCV. These bits are only used in OUTPUTCOMPARE or PWM mode and are cleared when CCMODE is written to 0b00 (Off).			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	SYNCBUSY	0x0	R	**Sync Busy**
	Indicates synchronization ongoing			
5	DTILOCKSTATUS	0x0	R	**DTI lock status**
	Indicates current status of DTI lock			

Value	Mode	Description
0	UNLOCKED	DTI registers are unlocked
1	LOCKED	DTI registers are locked

4	TIMERLOCKSTATUS	0x0	R	**Timer lock status**
	Indicates current status of Timer lock			

Value	Mode	Description
0	UNLOCKED	TIMER registers are unlocked
1	LOCKED	TIMER registers are locked

3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	TOPBV	0x0	R	**TOP Buffer Valid**
	This indicates that TIMERn_TOPB contains valid data that has not been written to TIMERn_TOP. This bit is also cleared when TIMERn_TOP is written.			
1	DIR	0x0	R	**Direction**
	Indicates count direction.			

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	UP		Counting up
	1	DOWN		Counting down
0	RUNNING	0x0	R	**Running**
	Indicates if timer is running or not.			

19.5.6 TIMER_IF - Interrupt Flag Register

Offset	Bit Position																														
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset						0x0	0x0	0x0		0x0	0x0	0x0		0x0	0x0	0x0										0x0	0x0	0x0		0x0	0x0	0x0
Access						RW	RW	RW		RW	RW	RW		RW	RW	RW										RW	RW	RW		RW	RW	RW
Name						ICFUF2	ICFUF1	ICFUF0		ICFOF2	ICFOF1	ICFOF0		ICFWLFULL2	ICFWLFULL1	ICFWLFULL0										CC2	CC1	CC0		DIRCHG	UF	OF

Bit	Name	Reset	Access	Description
31:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	ICFUF2	0x0	RW	**Input capture FIFO underflow**
	Indicates that software tried to read an empty FIFO on channel 2.			
25	ICFUF1	0x0	RW	**Input capture FIFO underflow**
	Indicates that software tried to read an empty FIFO on channel 1.			
24	ICFUF0	0x0	RW	**Input capture FIFO underflow**
	Indicates that software tried to read an empty FIFO on channel 0.			
23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22	ICFOF2	0x0	RW	**Input Capture FIFO overflow**
	Indicates that input capture FIFO for channel 2 has overflown, and a prior captured value was lost. The latest captured value can be read from the ICOF register.			
21	ICFOF1	0x0	RW	**Input Capture FIFO overflow**
	Indicates that input capture FIFO for channel 1 has overflown, and a prior captured value was lost. The latest captured value can be read from the ICOF register.			
20	ICFOF0	0x0	RW	**Input Capture FIFO overflow**
	Indicates that input capture FIFO for channel 0 has overflown, and a prior captured value was lost. The latest captured value can be read from the ICOF register.			
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	ICFWLFULL2	0x0	RW	**Input Capture Watermark Level Full**
	This bit indicates that the Input capture FIFO watermark for channel 2 has been exceeded.			
17	ICFWLFULL1	0x0	RW	**Input Capture Watermark Level Full**
	This bit indicates that the Input capture FIFO watermark for channel 1 has been exceeded.			
16	ICFWLFULL0	0x0	RW	**Input Capture Watermark Level Full**
	This bit indicates that the Input capture FIFO watermark for channel 0 has been exceeded.			
15:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	CC2	0x0	RW	**Capture Compare Channel 2 Interrupt Flag**

Bit	Name	Reset	Access	Description
	In INPUT CAPTURE mode this bit indicates that a new Capture event has taken place. In OUTPUTCOMPARE or PWM mode this bit indicates that a match event has taken place			
5	CC1	0x0	RW	**Capture Compare Channel 1 Interrupt Flag**
	In INPUT CAPTURE mode this bit indicates that a new Capture event has taken place. In OUTPUTCOMPARE or PWM mode this bit indicates that a match event has taken place			
4	CC0	0x0	RW	**Capture Compare Channel 0 Interrupt Flag**
	In INPUT CAPTURE mode this bit indicates that a new Capture event has taken place. In OUTPUTCOMPARE or PWM mode this bit indicates that a match event has taken place			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DIRCHG	0x0	RW	**Direction Change Detect Interrupt Flag**
	This bit is set when count direction changes. Set only in Quadrature Decoder mode			
1	UF	0x0	RW	**Underflow Interrupt Flag**
	This bit indicates that there has been an underflow.			
0	OF	0x0	RW	**Overflow Interrupt Flag**
	This bit indicates that there has been an overflow.			

19.5.7 TIMER_IEN - Interrupt Enable Register

Offset	Bit Position																														
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset						0x0	0x0	0x0		0x0	0x0	0x0		0x0	0x0	0x0										0x0	0x0	0x0		0x0	0x0	0x0
Access						RW	RW	RW		RW	RW	RW		RW	RW	RW										RW	RW	RW		RW	RW	RW
Name						ICFUF2	ICFUF1	ICFUF0		ICFOF2	ICFOF1	ICFOF0		ICFWLFULL2	ICFWLFULL1	ICFWLFULL0										CC2	CC1	CC0		DIRCHG	UF	OF

Bit	Name	Reset	Access	Description
31:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	ICFUF2	0x0	RW	**ICFUF2 Interrupt Enable**
	Enable/Disable the ICFUF2 interrupt			
25	ICFUF1	0x0	RW	**ICFUF1 Interrupt Enable**
	Enable/Disable the ICFUF1 interrupt			
24	ICFUF0	0x0	RW	**ICFUF0 Interrupt Enable**
	Enable/Disable the ICFUF0 interrupt			
23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22	ICFOF2	0x0	RW	**ICFOF2 Interrupt Enable**
	Enable/Disable the ICFOF2 interrupt			
21	ICFOF1	0x0	RW	**ICFOF1 Interrupt Enable**
	Enable/Disable the ICFOF1 interrupt			
20	ICFOF0	0x0	RW	**ICFOF0 Interrupt Enable**
	Enable/Disable the ICFOF0 interrupt			
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	ICFWLFULL2	0x0	RW	**ICFWLFULL2 Interrupt Enable**
	Enable/Disable the ICFWLFULL2 interrupt			
17	ICFWLFULL1	0x0	RW	**ICFWLFULL1 Interrupt Enable**
	Enable/Disable the ICFWLFULL1 interrupt			
16	ICFWLFULL0	0x0	RW	**ICFWLFULL0 Interrupt Enable**
	Enable/Disable the ICFWLFULL0 interrupt			
15:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	CC2	0x0	RW	**CC2 Interrupt Enable**
	Enable/Disable the CC2 interrupt			
5	CC1	0x0	RW	**CC1 Interrupt Enable**

Bit	Name	Reset	Access	Description
	Enable/Disable the CC1 interrupt			
4	CC0	0x0	RW	**CC0 Interrupt Enable**
	Enable/Disable the CC0 interrupt			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DIRCHG	0x0	RW	**Direction Change Detect Interrupt Enable**
	Enable/Disable the DIRCHG interrupt			
1	UF	0x0	RW	**Underflow Interrupt Enable**
	Enable/Disable the UF interrupt			
0	OF	0x0	RW	**Overflow Interrupt Enable**
	Enable/Disable the OF interrupt			

19.5.8 TIMER_TOP - Counter Top Value Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xFFFF																															
Access	RW																															
Name	TOP																															

Bit	Name	Reset	Access	Description
31:0	TOP	0xFFFF	RW	**Counter Top Value**
	These bits hold the TOP value for the counter			

19.5.9 TIMER_TOPB - Counter Top Value Buffer Register

Offset	Bit Position																													
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	TOPB																															

Bit	Name	Reset	Access	Description
31:0	TOPB	0x0	RW	**Counter Top Buffer Register**
	These bits hold the TOP buffer value.			

19.5.10 TIMER_CNT - Counter Value Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	CNT																															

Bit	Name	Reset	Access	Description
31:0	CNT	0x0	RW	**Counter Value**
	These bits hold the counter value.			

19.5.11 TIMER_LOCK - TIMER Configuration Lock Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	W															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x0	W	**Timer Lock Key**
	Write any other value than the unlock code to lock TIMERn_CTRL, TIMERn_CFG, TIMERn_CMD, TIMERn_TOP, TIMERn_CNT, TIMERn_CCx_CTRL, TIMERn_CCx_CFG, and TIMERn_CCx_OC from editing. Write the unlock code to unlock these registers.			

Value	Mode	Description
52864	UNLOCK	Write to unlock TIMER registers

19.5.12 TIMER_EN - module en

Offset	Bit Position																														
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**Timer Module Enable**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

19.5.13 TIMER_CCx_CFG - CC Channel Configuration Register

Offset	Bit Position																															
0x060	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x0	0x0	0x0	0x0														0x0			0x0	
Access											RW	RW	RW	RW														RW			RW	
Name											ICFWL	FILT	PRSCONF	INSEL														COIST			MODE	

Bit	Name	Reset	Access	Description
31:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	ICFWL	0x0	RW	**Input Capture FIFO watermark level**
	Sets the watermark level for generation of the ICFWLFULL interrupt and DMA requests. ICFWLFULL will be set and DMA requests may be generated if the number of free FIFO entries is less than or equal to ICFWL.			
20	FILT	0x0	RW	**Digital Filter**
	Enable digital filter.			

Value	Mode	Description
0	DISABLE	Digital Filter Disabled
1	ENABLE	Digital Filter Enabled

Bit	Name	Reset	Access	Description
19	PRSCONF	0x0	RW	**PRS Configuration**
	Select PRS pulse or level for PRS output.			

Value	Mode	Description
0	PULSE	Each CC event will generate a one EM01GRPACLK cycle high pulse
1	LEVEL	The PRS channel will follow CC out

Bit	Name	Reset	Access	Description
18:17	INSEL	0x0	RW	**Input Selection**
	Select Compare/Capture channel input.			

Value	Mode	Description
0	PIN	TIMERnCCx pin is selected
1	PRSSYNC	Synchornous PRS selected
2	PRSASYNCLEVEL	Asynchronous Level PRS selected
3	PRSASYNCPULSE	Asynchronous Pulse PRS selected

Bit	Name	Reset	Access	Description
16:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	COIST	0x0	RW	**Compare Output Initial State**

Bit	Name	Reset	Access	Description
	This bit is only used in Output Compare and PWM mode. When this bit is set in Compare or PWM mode,the output is set high when the counter is disabled. When counting resumes, this value will represent the initial value for the output. If the bit is cleared, the output will be cleared when the counter is disabled.			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	MODE	0x0	RW	**CC Channel Mode**

These bits select the mode for Compare/Capture channel.

Value	Mode	Description
0	OFF	Compare/Capture channel turned off
1	INPUTCAPTURE	Input Capture
2	OUTPUTCOMPARE	Output Compare
3	PWM	Pulse-Width Modulation

19.5.14 TIMER_CCx_CTRL - CC Channel Control Register

Offset	Bit Position																													
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0		0x0												0x0		0x0		0x0							0x0		
Access					RW		RW												RW		RW		RW							RW		
Name					ICEVCTRL		ICEDGE												CUFOA		COFOA		CMOA							OUTINV		

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:26	ICEVCTRL	0x0	RW	**Input Capture Event Control**

These bits control when a Compare/Capture PRS output pulse and interrupt flag is set. DMA request however is set on every capture.

Value	Mode	Description
0	EVERYEDGE	PRS output pulse and interrupt flag set on every capture
1	EVERYSECONDEDGE	PRS output pulse and interrupt flag set on every second capture
2	RISING	PRS output pulse and interrupt flag set on rising edge only (if ICEDGE = BOTH)
3	FALLING	PRS output pulse and interrupt flag set on falling edge only (if ICEDGE = BOTH)

25:24	ICEDGE	0x0	RW	**Input Capture Edge Select**

These bits control which edges the edge detector triggers on. The output is used for input capture and external clock input.

Value	Mode	Description
0	RISING	Rising edges detected
1	FALLING	Falling edges detected
2	BOTH	Both edges detected
3	NONE	No edge detection, signal is left as it is

23:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	CUFOA	0x0	RW	**Counter Underflow Output Action**

Select output action on counter underflow.

Value	Mode	Description
0	NONE	No action on counter underflow
1	TOGGLE	Toggle output on counter underflow
2	CLEAR	Clear output on counter underflow
3	SET	Set output on counter underflow

Bit	Name	Reset	Access	Description
11:10	COFOA	0x0	RW	**Counter Overflow Output Action**
	Select output action on counter overflow.			

Value	Mode	Description
0	NONE	No action on counter overflow
1	TOGGLE	Toggle output on counter overflow
2	CLEAR	Clear output on counter overflow
3	SET	Set output on counter overflow

9:8	CMOA	0x0	RW	**Compare Match Output Action**
	Select output action on compare match.			

Value	Mode	Description
0	NONE	No action on compare match
1	TOGGLE	Toggle output on compare match
2	CLEAR	Clear output on compare match
3	SET	Set output on compare match

7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	OUTINV	0x0	RW	**Output Invert**
	Setting this bit inverts the output from the CC channel (Output compare or PWM mode).			
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

19.5.15 TIMER_CCx_OC - OC Channel Value Register

Offset	Bit Position																															
0x068	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	OC																															

Bit	Name	Reset	Access	Description
31:0	OC	0x0	RW	**Output Compare Value**
	This fields holds the output compare value			

19.5.16 TIMER_CCx_OCB - OC Channel Value Buffer Register

Offset	Bit Position																															
0x070	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	OCB																															

Bit	Name	Reset	Access	Description
31:0	OCB	0x0	RW	**Output Compare Value Buffer**
	This field holds the Output Compare buffer value which will be written to TIMERn_CCx_OC on an update event if TIMERn_CCx_OCB contains valid data			

19.5.17 TIMER_CCx_ICF - IC Channel Value Register

Offset	Bit Position																															
0x074	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R(r)																															
Name	ICF																															

Bit	Name	Reset	Access	Description
31:0	ICF	0x0	R(r)	**Input Capture FIFO**
	This FIFO holds captured values in input capture mode. Reading this register will pop the oldest unread value from the FIFO.			

19.5.18 TIMER_CCx_ICOF - IC Channel Value Overflow Register

Offset	Bit Position																															
0x078	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	ICOF																															

Bit	Name	Reset	Access	Description
31:0	ICOF	0x0	R	**Input Capture FIFO Overflow**
	This register always contains the most recent input capture value. If the input capture FIFO is full and a new capture occurs, this register will be updated and the previous caputre value is over-written.			

19.5.19 TIMER_DTCFG - DTI Configuration Register

Offset	Bit Position																													
0x0E0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0x0	0x0	0x0								0x0	0x0
Access																					RW	RW	RW								RW	RW
Name																					DTPRSEN	DTFATS	DTAR								DTDAS	DTEN

Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	DTPRSEN	0x0	RW	**DTI PRS Source Enable**
	Enable/disable PRS as DTI input.			
10	DTFATS	0x0	RW	**DTI Fault Action on Timer Stop**
	When Timer stops, DTI block outputs go to safe state as programmed in DTFA field of TIMERn_DTFC register. However, when DTAR is also set,DTAR having higher priority allows channel0 to output the incoming PRS input while the other channels go to safe state			
9	DTAR	0x0	RW	**DTI Always Run**
	This is used only for DTI channel 0. It Allows DTI channel 0 to keep running even when the timer is stopped. This is useful when its input source is PRS. However, here the undivided peripheral clock is always used regardless of the programmed value in DTPRESC.			
8:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	DTDAS	0x0	RW	**DTI Automatic Start-up Functionality**
	Configure DTI restart on debugger exit.			

Value	Mode	Description
0	NORESTART	No DTI restart on debugger exit
1	RESTART	DTI restart on debugger exit

0	DTEN	0x0	RW	**DTI Enable**
	Enable/disable DTI.			

19.5.20 TIMER_DTTIMECFG - DTI Time Configuration Register

Offset	Bit Position																															
0x0E4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x0						0x0						0x0									
Access											RW						RW						RW									
Name											DTFALLT						DTRISET						DTPRESC									

Bit	Name	Reset	Access	Description
31:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:16	DTFALLT	0x0	RW	**DTI Fall-time**
	Set time span for the falling edge. The fall time is DTFALLT+1 prescaled peripheral clock cycles			
15:10	DTRISET	0x0	RW	**DTI Rise-time**
	Set time span for the rising edge. The rise time is DTRISET+1 prescaled peripheral clock cycles			
9:0	DTPRESC	0x0	RW	**DTI Prescaler Setting**
	These bits select the prescaling factor for DTI. The selected timer clock will be divided by DTPRESC+1 before clocking the DTI logic.			

19.5.21 TIMER_DTFCFG - DTI Fault Configuration Register

Offset	Bit Position																															
0x0E8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0	0x0	0x0	0x0	0x0							0x0																	
Access				RW	RW	RW	RW	RW							RW																	
Name				DTEM23FEN	DTLOCKUPFEN	DTDBGFEN	DTPRS1FEN	DTPRS0FEN							DTFA																	

Bit	Name	Reset	Access	Description
31:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28	DTEM23FEN	0x0	RW	**DTI EM23 Fault Enable**
	Set this bit to 1 to enable EM2 or EM3 entry as a fault source			
27	DTLOCKUPFEN	0x0	RW	**DTI Lockup Fault Enable**
	Set this bit to 1 to enable core lockup as a fault source			
26	DTDBGFEN	0x0	RW	**DTI Debugger Fault Enable**
	Set this bit to 1 to enable debugger as a fault source			
25	DTPRS1FEN	0x0	RW	**DTI PRS 1 Fault Enable**
	Set this bit to 1 to enable PRS source 1 as a fault source			
24	DTPRS0FEN	0x0	RW	**DTI PRS 0 Fault Enable**
	Set this bit to 1 to enable PRS source 0 as a fault source			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	DTFA	0x0	RW	**DTI Fault Action**
	Select fault action.			

Value	Mode	Description
0	NONE	No action on fault
1	INACTIVE	Set outputs inactive
2	CLEAR	Clear outputs
3	TRISTATE	Tristate outputs

15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

19.5.22 TIMER_DTCTRL - DTI Control Register

Offset	Bit Position																													
0x0EC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															DTIPOL	DTCINV

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	DTIPOL	0x0	RW	**DTI Inactive Polarity**
	Set inactive polarity of outputs			
0	DTCINV	0x0	RW	**DTI Complementary Output Invert.**
	DTI Complementary Output Invert.			

19.5.23 TIMER_DTOGEN - DTI Output Generation Enable Register

Offset	Bit Position																													
0x0F0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											RW	RW	RW	RW	RW	RW
Name																											DTOGCDTI2EN	DTOGCDTI1EN	DTOGCDTI0EN	DTOGCC2EN	DTOGCC1EN	DTOGCC0EN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	DTOGCDTI2EN	0x0	RW	**DTI CDTIn Output Generation Enable**
	This bit enables/disables output generation for the CDTI output from the DTI.			
4	DTOGCDTI1EN	0x0	RW	**DTI CDTIn Output Generation Enable**
	This bit enables/disables output generation for the CDTI output from the DTI.			
3	DTOGCDTI0EN	0x0	RW	**DTI CDTIn Output Generation Enable**
	This bit enables/disables output generation for the CDTI output from the DTI.			
2	DTOGCC2EN	0x0	RW	**DTI CCn Output Generation Enable**
	This bit enables/disables output generation for the CC output from the DTI.			
1	DTOGCC1EN	0x0	RW	**DTI CCn Output Generation Enable**
	This bit enables/disables output generation for the CC output from the DTI.			
0	DTOGCC0EN	0x0	RW	**DTI CCn Output Generation Enable**
	This bit enables/disables output generation for the CC output from the DTI.			

19.5.24 TIMER_DTFAULT - DTI Fault Register

Offset	Bit Position																													
0x0F4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												R	R	R	R	R
Name																												DTEM23F	DTLOCKUPF	DTDBGF	DTPRS1F	DTPRS0F

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	DTEM23F	0x0	R	**DTI EM23 Entry Fault**
	This bit is set to 1 if EM2 or EM3 entry has occurred and DTEM23FEN is set to 1. The TIMER0_DTFAULTC register can be used to clear fault bits.			
3	DTLOCKUPF	0x0	R	**DTI Lockup Fault**
	This bit is set to 1 if a core lockup fault has occurred and DTLOCKUPFEN is set to 1. The TIMER0_DTFAULTC register can be used to clear fault bits.			
2	DTDBGF	0x0	R	**DTI Debugger Fault**
	This bit is set to 1 if a debugger fault has occurred and DTDBGFEN is set to 1. The TIMER0_DTFAULTC register can be used to clear fault bits.			
1	DTPRS1F	0x0	R	**DTI PRS 1 Fault**
	This bit is set to 1 if a PRS 1 fault has occurred and DTPRS1FEN is set to 1. The TIMER0_DTFAULTC register can be used to clear fault bits.			
0	DTPRS0F	0x0	R	**DTI PRS 0 Fault**
	This bit is set to 1 if a PRS 0 fault has occurred and DTPRS0FEN is set to 1. The TIMER0_DTFAULTC register can be used to clear fault bits.			

19.5.25 TIMER_DTFAULTC - DTI Fault Clear Register

Offset	Bit Position																													
0x0F8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												W	W	W	W	W
Name																												DTEM23FC	DTLOCKUPFC	DTDBGFC	DTPRS1FC	DTPRS0FC

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	DTEM23FC	0x0	W	**DTI EM23 Fault Clear**
	Write 1 to this bit to clear EM23 entry fault.			
3	DTLOCKUPFC	0x0	W	**DTI Lockup Fault Clear**
	Write 1 to this bit to clear core lockup fault.			
2	DTDBGFC	0x0	W	**DTI Debugger Fault Clear**
	Write 1 to this bit to clear debugger fault.			
1	DTPRS1FC	0x0	W	**DTI PRS1 Fault Clear**
	Write 1 to this bit to clear PRS 1 fault.			
0	DTPRS0FC	0x0	W	**DTI PRS0 Fault Clear**
	Write 1 to this bit to clear PRS 0 fault.			

19.5.26 TIMER_DTLOCK - DTI Configuration Lock Register

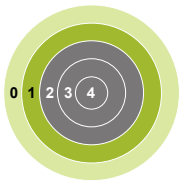
Offset	Bit Position																													
0x0FC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									DTILOCKKEY							

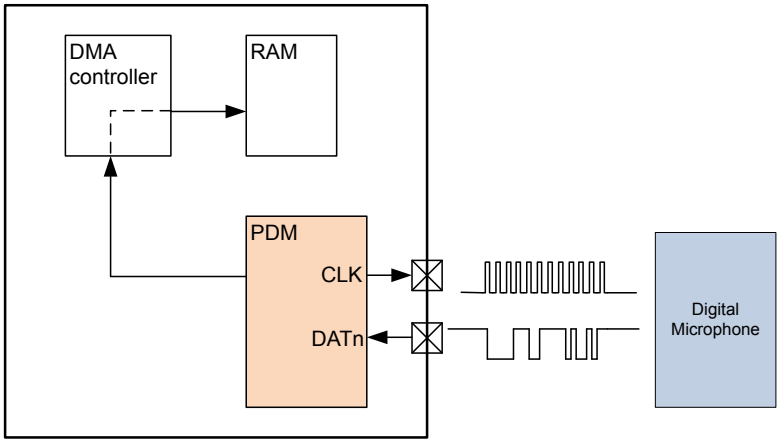
Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	DTILOCKKEY	0x0	W	**DTI Lock Key**

Write any other value than the unlock code to lock TIMER_ROUTE, TIMER_DTCTRL, TIMER_DTCFG, TIMER_DTTIMECFG and TIMER_DTFCFG from editing. Write the unlock code to unlock the DTI registers.

Value	Mode	Description
52864	UNLOCK	Write to unlock TIMER DTI registers

20. PDM - PDM Interface





Quick Facts

What?

The PDM Module accepts a PDM bitstream input and provides PCM output samples.

Why?

Peripherals with PDM digital output signals reduce system cost and provide low-cost isolation.

How?

The PDM module uses a Programmable-Order CIC decimation filter to reduce the high-speed single bit input to a lower speed multi-bit wide PCM sample.

20.1 Introduction

The PDM module provides a decimation filter for Pulse Density Modulation (PDM) microphones, isolated Sigma-Delta ADCs, and other PDM or Sigma-Delta bit stream peripherals.

The decimation filter consists of a Cascaded Integrator Comb (CIC) filter. The output width of the comb filter is selectable, supporting 16, 24, or 32 bits. The comb filter has a fixed M value of 1.

20.2 Features

- Cascaded Integrator Comb Filter
- Programmable Filter Order
  - 2, 3, 4, or 5
- Programmable Down Sample Rate
  - Power of 2 Rate
  - Integer Rate
- Selectable Output Width
  - 16-bit output data
  - 24-bit output data
  - 32-bit output data
- Selectable Data Alignment (16-bit or 24-bit mode)
  - Left Aligned Data
  - Right Aligned Data
- 32-bit raw Mode - Data Directly from Integrator
- Programmable Gain
  - Supports Integer Down Sample Rates
  - Supports dynamic gain changes
  - 6 dB steps
- Supports Multiple Channels
  - 1 or 2 channels
- Supports Stereo Input Data
- Data FIFO
  - Programmable Data Valid Level
  - 16-bit packed mode
- DMA support
  - DMA request on Data Valid
- Interrupts
  - Data Valid
  - Overflow
- PRS Output on Down Sample Rate clock
- Clock Prescaler
- Applications
  - Sigma-Delta Modulators
  - Isolated Sigma Delta Modulators
  - PDM Microphones
  - Digital Sensors

20.3 Functional Description

20.3.1 Overview

An overview of PDM is shown in Figure 20.1 PDM Overview on page 539.

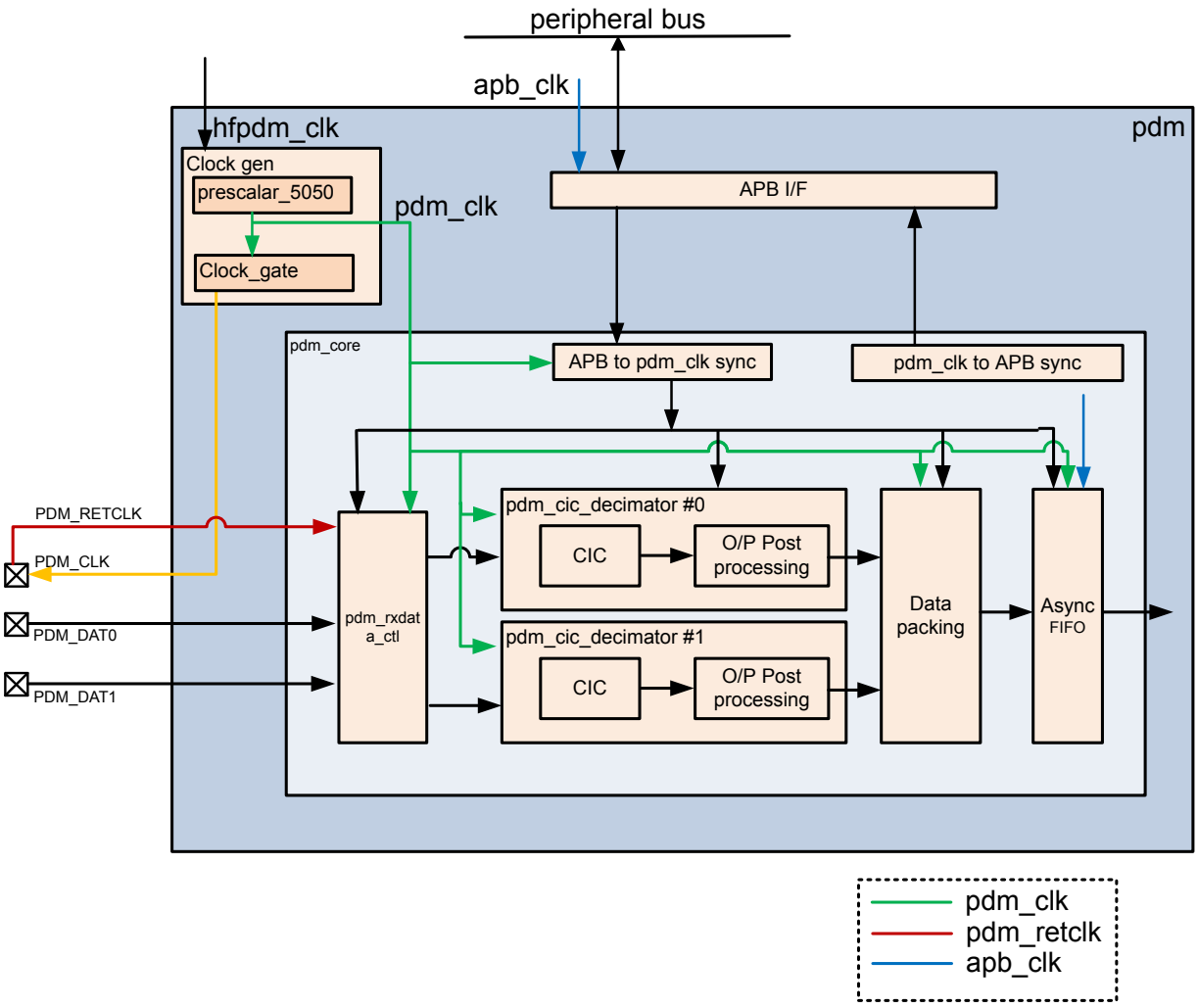


Figure 20.1. PDM Overview

20.3.2 PDM Clock Generation

The module generates a PDM output clock using an integer divider prescaler to divide down the module HFCLK_PDM. See 8. CMU - Clock Management Unit for details on generation of the HFCLK_PDM.

The PDM module includes a 10-bit integer prescaler. The PDM clock is divided by PRESC plus 1. The PRESC field is located in the PDMn_CFG1 register.

For optimum performance, PDM signal sources require an accurate, jitter-free clock.

20.3.3 Filter Order

The decimation filter order is programmable from 2 to 5. The selected filter order must be at least one order higher than the sigma delta modulator. Isolated ADCs typically have a second order sigma delta modulator and require a 3rd order filter. PDM microphones typically have a fourth order modulator and require a 5th order filter. The filter also supports 1st order and 3rd order sigma delta modulators.

The filter order is programmed in the FORDER field of the PDMn_CFG0 register. The filter order must be programmed before enabling the module and should not be modified while the module is running.

20.3.4 Down Sample Rate

The Down Sample Rate of the decimation filter is programmable. The decimation filter supports both power of 2 (2x) and integer down sampling rates. The advantage of using a power of two down sampling rate is that gain compensation requires only shifting the results and the full scale value will use all available bits. The benefit of using non-power of two down sampling rates is that this allows finer adjustment of the output word rate.

The down sample rate is programmed into the DSR field of the PDMn_CTRL register and can be changed dynamically while the filter is running. The changed DSR setting will be loaded to the dsr counter when the dsr counter rolls over to zero.

The decimation filter includes gain compensation which will shift the final results so that a full scale output is achieved for fully modulated input. When filter output option is 16-bit or 24-bit then program the shift value in gain field of CTRL register. Gain = 32 - (1 + ceiling(((log10(G)/log10(2)),1). Where G = (DSR ^ N).

The GAIN field is in the PDMn_CTRL register and can be changed dynamically while the filter is running. The value of the GAIN field can vary dynamically from 0 to a maximum value of the integer part of the LOG2 of the down sample rate.

The maximum down sample rate is limited by the internal width of the integrator. The maximum value depends on the filter order.

Table 20.1. Maximum DSR vs Filter Order

N	Max DSR
3	1290
4	215
5	73

20.3.5 Multi Channel Operation

The PDM module has 2 channels. All channels use a common clock and filter settings. The channels are synchronized such that all channels down sample and produce output data on the same filter clock cycle.

The module supports clock output mode only. There is only one clock output for all channels.

The number of active channels is selectable - 1 or 2 channels.

In normal mode, there is one clock output and one data input for each channel.

The two channel stereo mode samples DAT0 on both rising and falling edges of the clock, to produce data output for CH0 and CH1. CH0 uses the rising edge data, and CH1 uses the falling edge data.

In normal mode, data is normally clocked on the rising edge of the CLK signal. There is an option to invert the data polarity which will clock data on the falling edge of the clock signal. This may be useful for some sigma delta modulators that have insufficient hold time on the rising clock edge.

20.3.6 Output Options

The output width of the comb filter is selectable, supporting 16, 24, or 32 bits. When 32-bit data is selected, all 32-bits will be significant. There are options for right or left justified 16-bit or 24-bit data.

There is also an option for raw 32-bit data directly from the integrator. This option supports a software comb filter.

20.3.7 FIFO

The PDM module has a 32-bit by four entry FIFO. The FIFO has the capacity to store up to eight 16-bit samples, four 32-bit samples, or four 24-bit samples. Each sample is normally stored as a 32-bit word, unless the DATAFORMAT field is set to DOUBLE16 in the PDMn_CFG0 register. The PDM module stores data in the FIFO starting with CH0. The DMA or software should read out all samples for all enabled channels starting with CH0.

The FIFO has a programmable Data Valid Level interrupt, FIFO full interrupt, and FIFO overflow interrupt.

If the DATAFORMAT field is set to DOUBLE16, then 16-bit data from two channels are packed into a single 32-bit word and written into the FIFO.

If NUMCH bitfield is 0 (i.e. only one channel, CH0) and the DATAFORMAT field is set to DOUBLE16, then the UPPER16BIT are zeros and LOWER16BIT data has CH0 16-bit data.

20.3.8 DMA Support

The module will generate a DMA request when the number of samples in the FIFO is equal to the level set by the FIFODVL field in the PDMn_CFG0 register.

20.3.9 PRS Support

The module has a PRS sync output which generates a PRS output pulse on the onset of the down sample rate clock. This PRS signal may be used to trigger an ADC conversion, routed to a timer/counter, PCNT, or to a GPIO pin.

20.3.10 PDM Energy Modes

- The PDM interface functions in EM0/EM1 mode.
- All the PDM registers are retained in EM2/EM3; after waking from EM2/EM3 the system can resume using the PDM peripheral without reconfiguring it.
- The PDM interrupt can be used as as wakeup source from EM1.
- The following are steps the software must perform for EM2/EM3 entry and exit:
  - Read any converted PCM samples which are available in the FIFO.
  - Issue the Filter STOP command by configuring PDMn_CMD register. This will gate off the clock to the external device.
  - Poll until the ACT bit in the STATUS register is low. ACT indicates the filter status to software.
  - Issue a WFI command to enter EM2/EM3.
  - Upon EM2/EM3 exit, clear the filter by issuing CLEAR command in PDMn_CMD register.
  - Flush the FIFO by issuing FIFOFL command in PDMn_CMD register.
  - To restart, issue the Filter START command by configuring PDMn_CMD register.

20.3.11 Debug Mode

When CPU is halted in debug mode, the PDM will continue to run and DMA will capture the data from the FIFO.

20.3.12 Pin Configurations

A common GPIO pin is used as the clock output for up to 2 DATA ports.

Three route locations are available via DBUS and the DBUS connections are detailed in the device data sheet. Pin locations can be selected using the GPIO_PDM_CLKROUTE and GPIO_PDM_DATxROUTE registers. The clock output is enabled using the GPIO_PDM_ROUTEEN register. The slew rate for the CLK pin should be set to its maximum (fastest) setting.

All pins should be selected from the same DBUS (DBUSAB or DBUSCD). Optimal performance is achieved when all pins are routed adjacent to each other on the same GPIO port (PA, PB, PC, or PD).

20.3.13 Programmer's Model

- The PDM configuration registers are considered to be static and can only be updated when the filter is not running. Only DSR/Gain can be changed dynamically.
- Stop the filter before changing the prescale value. Stopping the filter will gate off the clock to external devices. Before re-starting the filter, issue a Filter CLEAR command "PDMn_CMD.CLEAR" and issue a FIFO Flush command "PDMn_CMD.FIFOFL". The Filter CLEAR command acts as soft reset to the Filter and the Flush command as soft reset to the FIFO pointers.
- When there is an Overflow error condition, without Stopping the filter, issue a FIFO Flush command.

  When a FLUSH command is issued, poll for FLUSHFLBUSY, don't read STATUS bits when FLUSHFLBUSY is high.

  Before issuing FLUSH command clear any pending DMA requests.
- The APB clock and PDM Core clock are asynchronous to each other, so when a FIFOFL command is issued, the FIFOFLBUSY bit in SYNCBUSY should be polled. For START/STOP/CLEAR commands, the SYNCBUSY bit should be polled.

  Similarly when the DSR/GAIN fields in CTRL register are configured, poll for SYNCBUSY bit in the SYNCBUSY register.
- The START command is common for all channels of the filter. When NUMCH is set to 1, i.e. when two channels are configured, both channel filter outputs are available at a time and hardware will take 6 cycles to write into FIFO, so there is a limitation on the minimum DSR setting.
- If the DATAFORMAT selected in PDMn_CFG0 is 32-bit/24-bit/16-bit, then Minimum DSR > (NUMCH + 3).

  Example: If NUMCH is 1 (two channels), then Minimum DSR should be 4.
- If the DATAFORMAT selected in PDMn_CFG0 is DOUBLE16, then Minimum DSR > ((NUMCH + 1)/2 + 2).

  Example: If NUMCH is 1 (two channels) and the DATAFORMAT is DOUBLE16, then Minimum DSR should be 3.

20.3.13.1 Configuration example 1

Sample program for configuring PDM

- Enable APB clocks in CMU, configure PDM clock in CMU

```
/* Configure GPIO clock, LDMA clock */
CMU->CLKEN0 = CMU->CLKEN0 | CMU_CLKEN0_GPIO | CMU_CLKEN0_LDMA;
/* Enable PCLK of PDM peripheral */
CMU->CLKEN0 = (CMU->CLKEN0 | CMU_CLKEN0_PDM);
/* If required choose PER clock prescale value */
CMU->SYSCLKCTRL = (prescale_value << _CMU_SYSCLKCTRL_PCLKPRESC_SHIFT);
```

Select PDM clock (from EM01GRPBCLK)

```
CMU->EM01GRPBCLK = CMU_EM01GRPBCLK_CLKSEL_HFRCODPLL;
```

- Configure GPIOs using GPIO_PDM_ROUTEEN/CLKROUTE/DAT0ROUTE/DAT1ROUTE registers

  Set CLK to PUSHPULL mode and configure SLEWRATE to 7

  Set DAT* PIN to Input mode
- Based on selected clock source, configure the "PRESCALE" value of PDM

```
while(PDM->SYNCBUSY !=0);
PDM->CFG1 = (prescale_val << _PDM_CFG1_PRESC_SHIFT);
```

- Configure PDM

  In this example PDM is configured to single channel, data capture polarity of channel is set 0, filter output to 32-bit, FIFO data valid level to max (3), DSR to 64 and Gain to 0, Filter order set to 5

```
PDM->CFG0 = (0 << _PDM_CFG0_NUMCH_SHIFT |
1 << _PDM_CFG0_CHOCLKPOL_SHIFT |
3 << _PDM_CFG0_DATAFORMAT_SHIFT |
3 << _PDM_CFG0_FORDER_SHIFT);
```

Configure DSR/Gain, in this example DSR is set to decimal 64, since 32-bit output no need to shift, so Gain is set to 0

```
while(PDM->SYNCBUSY !=0);
PDM->CTRL = ((0 << _PDM_CTRL_GAIN_SHIFT) | (64 << _PDM_CTRL_DSR_SHIFT));
```

- Configure PDM interrupts
- Enable PDM module
  PDM->EN = PDM_EN_EN
- Configure LDMA to capture the Data
- Start the Filter

```
while(PDM->SYNCBUSY !=0);
PDM->CMD = PDM_CMD_START;
```

- In ISR routine, If there is overflow or underflow is set, without stopping the filter issue the FLUSH command

```
while(PDM->SYNCBUSY !=0);
PDM->CMD = PDM_CMD_FIFOFL;
```

- After the desired number of samples are collected, if PDM need to be stopped then

  Stop PDM filter

```
while(PDM->SYNCBUSY !=0);
PDM->CMD = PDM_CMD_STOP;
```

  CLEAR PDM filter

```
while(PDM->SYNCBUSY !=0);
PDM->CMD = PDM_CMD_CLEAR;
```

  FLUSH PDM FIFO

```
while(PDM->SYNCBUSY !=0);
PDM->CMD = PDM_CMD_FIFOFL;
```

  Disable PDM module

20.4 Applications

20.4.1 PDM Microphones

Micro-Electrical Mechanical (MEMS) microphones offer lower cost, smaller packages, and better reliability than electret microphones. These microphones are available with analog or digital PDM outputs. The PDM version does not require an expensive audio quality ADC and offer a lower overall system cost. This PDM module works with most MEMS microphones with a PDM digital output.

Internal to the PDM microphone, an analog signal drives a sigma delta modulator and outputs a Pulse Density Modulated bit stream. The MCU provides a clock to the microphone and the microphone provides a signal back. Some PDM microphones support stereo using both edges of the clock. This eliminates one of the wires, which is important for some products when the wiring and/or connector cost are significant. It also frees up on additional GPIO pin on the MCU. Note that it is also possible to support stereo using one clock and two signal wires, one for right and one for left, this just uses more pins and more wire connects.

A high quality audio codec might use a 256x oversampling ratio with a 12.288 MHz clock for 48 kHz audio. The PDM microphones have a limited bandwidth for the oversampling clock. Most MEMS PDM microphones have a 1-3 MHz range. It is common to use exactly 3.072 MHz for 48 kHz audio.

Because of the limited oversampling clock range, the CIC filter for PDM microphones must have a lower oversampling ratio and a higher order than other applications. Most PDM microphones use a 4th order sigma delta modulator and require a 5th order decimation filter. The 5th order filter will provide a 16 bit ENOB with a 48 kHz sample rate and 1.536 MHz clock rate (32x OSR).

	F_{clk} (MHz)				ENOB vs order	
	3.072	2.8224	1.536	1.024		
OSR	F_{samp} (kHz)	F_{samp} (kHz)	F_{samp} (kHz)	F_{samp} (kHz)	4	5
16	192	176.4	96	64	10.7	13.3
32	96	88.2	48	32	14.2	17.8
64	48	44.1	24	16	17.7	22.3
128	24	22.05	12	8	21.2	-

20.4.2 Isolated Sigma Delta Modulators

The PDM module also supports isolated sigma delta modulators with a clock in and sigma delta output, such as the Silicon Labs Si8940.

The desired sample frequency depends on the chip limits of the isolated sigma delta converter, and also the application. Most isolated sigma delta modulators have a limit of 10 or 20 MHz. For industrial motor control, the common PWM and sample frequencies are 16 kHz, 12 kHz, and 8 kHz. Electric metering applications are required to measure the harmonics up to the 50th harmonic, and the sample rate must be at least twice this frequency. So the sample frequency must be at least 5 kHz for 50 Hz, or 6 kHz for 60 Hz. The table below summarizes the typical sampling frequencies.

A third order filter is generally sufficient for most isolated sigma delta applications. Most available sigma delta modulators are second order. There is no need to use higher than third order to achieve a high ENOB using a low clock rate, as with the PDM microphones. However, there is a need for higher decimation ratios to achieve the desired low sample frequencies. A max DSR of 256 will support near 24-bit performance with an output word rate of 10 to 20 kHz.

Most available isolated sigma delta modulators are limited to an ENOB of only 14 bits. So using a higher OSR will not necessarily result in a higher ENOB or signal to noise ration. However, using a higher OSR will result in a lower output word rate.

	F_{clk} (MHz)						ENOB vs order		
	Chip Limits		Motor Control		Metering				
	20	10	16.384	12.288	12.288	10.24			
OSR	F_{samp} (kHz)	F_{samp} (kHz)	F_{samp} (kHz)	F_{samp} (kHz)	F_{samp} (kHz)	F_{samp} (kHz)	3	4	5
16	1250.	625.	1024	768	768	640	8.2	10.7	13.3
32	625.	312.5	512	384	384	320	10.7	14.2	17.8
64	312.5	156.25	256	192	192	160	13.2	17.7	22.3
128	156.25	78.125	128	96	96	80	15.7	21.2	
256	78.125	39.063	64	48	48	40	18.2		
512	39.063	19.531	32	24	24	20	20.7		
1024	19.531	9.766	16	12	12	10	23.2		

20.5 PDM Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	PDM_IPVERSION	R	IP Version ID
0x004	PDM_EN	RW ENABLE	PDM Module enable Register
0x008	PDM_CTRL	RW SYNC	PDM Core Control Register
0x00C	PDM_CMD	W SYNC	PDM Core Command Register
0x010	PDM_STATUS	RH	PDM Status register
0x014	PDM_CFG0	RW CONFIG	PDM Core Configuration Register0
0x018	PDM_CFG1	RW CONFIG	PDM Core Configuration Register1
0x020	PDM_RXDATA	R(r)H	PDM Received Data Register
0x040	PDM_IF	RWH INTFLAG	Interrupt Flag Register
0x044	PDM_IEN	RW	Interrupt Flag Register
0x060	PDM_SYNCBUSY	RH	Synchronization Busy Register
0x1000	PDM_IPVERSION_SET	R	IP Version ID
0x1004	PDM_EN_SET	RW ENABLE	PDM Module enable Register
0x1008	PDM_CTRL_SET	RW SYNC	PDM Core Control Register
0x100C	PDM_CMD_SET	W SYNC	PDM Core Command Register
0x1010	PDM_STATUS_SET	RH	PDM Status register
0x1014	PDM_CFG0_SET	RW CONFIG	PDM Core Configuration Register0
0x1018	PDM_CFG1_SET	RW CONFIG	PDM Core Configuration Register1
0x1020	PDM_RXDATA_SET	R(r)H	PDM Received Data Register
0x1040	PDM_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1044	PDM_IEN_SET	RW	Interrupt Flag Register
0x1060	PDM_SYNCBUSY_SET	RH	Synchronization Busy Register
0x2000	PDM_IPVERSION_CLR	R	IP Version ID
0x2004	PDM_EN_CLR	RW ENABLE	PDM Module enable Register
0x2008	PDM_CTRL_CLR	RW SYNC	PDM Core Control Register
0x200C	PDM_CMD_CLR	W SYNC	PDM Core Command Register
0x2010	PDM_STATUS_CLR	RH	PDM Status register
0x2014	PDM_CFG0_CLR	RW CONFIG	PDM Core Configuration Register0
0x2018	PDM_CFG1_CLR	RW CONFIG	PDM Core Configuration Register1
0x2020	PDM_RXDATA_CLR	R(r)H	PDM Received Data Register
0x2040	PDM_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2044	PDM_IEN_CLR	RW	Interrupt Flag Register
0x2060	PDM_SYNCBUSY_CLR	RH	Synchronization Busy Register
0x3000	PDM_IPVERSION_TGL	R	IP Version ID
0x3004	PDM_EN_TGL	RW ENABLE	PDM Module enable Register

Offset	Name	Type	Description
0x3008	PDM_CTRL_TGL	RW SYNC	PDM Core Control Register
0x300C	PDM_CMD_TGL	W SYNC	PDM Core Command Register
0x3010	PDM_STATUS_TGL	RH	PDM Status register
0x3014	PDM_CFG0_TGL	RW CONFIG	PDM Core Configuration Register0
0x3018	PDM_CFG1_TGL	RW CONFIG	PDM Core Configuration Register1
0x3020	PDM_RXDATA_TGL	R(r)H	PDM Received Data Register
0x3040	PDM_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3044	PDM_IEN_TGL	RW	Interrupt Flag Register
0x3060	PDM_SYNCBUSY_TGL	RH	Synchronization Busy Register

20.6 PDM Register Description

20.6.1 PDM_IPVERSION - IP Version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP VERSION**
	Indicates the version of IP			

20.6.2 PDM_EN - PDM Module enable Register

Offset	Bit Position																														
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**PDM enable** when set to 1, module is enabled

Value	Mode	Description
0	DISABLE	Disable module
1	ENABLE	Enable module

20.6.3 PDM_CTRL - PDM Core Control Register

Offset	Bit Position																														
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0															0x0				
Access													RW															RW				
Name													DSR															GAIN				

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:8	DSR	0x0	RW	**Down sampling rate of Decimation filter** Down sampling rate, it can be an integer or power of 2 .
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4:0	GAIN	0x0	RW	**Selects Gain factor of DCF** Selects Gain factor of DCF .

20.6.4 PDM_CMD - PDM Core Command Register

Offset	Bit Position																														
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0								0x0				0x0				0x0
Access																W								W				W				W
Name																FIFOFL								CLEAR				STOP				START

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	FIFOFL	0x0	W	**FIFO Flush**
	one hot for selectively clearing the filter.			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	CLEAR	0x0	W	**Clear DCF**
	one hot for selectively clearing the filter.			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	STOP	0x0	W	**Stop DCF**
	one hot for selectively stoping the filter.			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	START	0x0	W	**Start DCF**
	one hot for selectively starting the filter.			

20.6.5 PDM_STATUS - PDM Status register

Offset	Bit Position																														
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x0					0x1	0x0				0x0
Access																						R					R	R				R
Name																						FIFOCNT					EMPTY	FULL				ACT

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	FIFOCNT	0x0	R	**FIFO CNT**
	Indicates number of valid data words in the PDM FIFO.			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	EMPTY	0x1	R	**FIFO EMPTY Status**
	Indicates PDM FIFO is Empty.			
4	FULL	0x0	R	**FIFO FULL Status**
	Indicates PDM FIFO is Full.			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	ACT	0x0	R	**PDM is active**
	Indicates PDM is running.			

20.6.6 PDM_CFG0 - PDM Core Configuration Register0

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset							0x0	0x0								0x0			0x0			0x0					0x0			0x0	
Access							RW	RW								RW			RW			RW					RW			RW	
Name							CH1CLKPOL	CH0CLKPOL								STEREOMODECH01			FIFODVL			DATAFORMAT					NUMCH			FORDER	

Bit	Name	Reset	Access	Description
31:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	CH1CLKPOL	0x0	RW	**CH1 CLK Polarity**
	Configure CH1 CLK Polarity.			
	Value	Mode		Description
	0	NORMAL		Input data clocked on rising clock edge.
	1	INVERT		Input data clocked on falling clock edge.
24	CH0CLKPOL	0x0	RW	**CH0 CLK Polarity**
	Configure CH0 CLK Polarity.			
	Value	Mode		Description
	0	NORMAL		Input data clocked on rising clock edge.
	1	INVERT		Input data clocked on falling clock edge.
23:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	STEREOMODECH01	0x0	RW	**Stereo mode CH01**
	Stereo mode Configuration for Channel pair CH0 and CH1.			
	Value	Mode		Description
	0	DISABLE		No Stereo mode.
	1	CH01ENABLE		CH0 and CH1 in Stereo mode.
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	FIFODVL	0x0	RW	**Data Valid level in FIFO**
	Configure FIFO Data valid level, water-mark.			
	Value	Mode		Description
	0	ONE		Atleast one word.

Bit	Name	Reset	Access	Description
	1	TWO		Two words.
	2	THREE		Three words.
	3	FOUR		Four words.
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	DATAFORMAT	0x0	RW	**Filter output format**
	Configure Filter data output format.			
	Value	Mode		Description
	0	RIGHT16		Right aligned 16-bit, left bits are sign extended.
	1	DOUBLE16		Pack two 16-bit samples into one 32-bit word.
	2	RIGHT24		Right aligned 24bit, left bits are sign extended.
	3	FULL32BIT		32 bit data.
	4	LEFT16		Left aligned 16-bit, right bits are zeros.
	5	LEFT24		Left aligned 24-bit, right bits are zeros.
	6	RAW32BIT		RAW 32 bit data from Integrator.
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	NUMCH	0x0	RW	**Number of Channels**
	Number of Channels.			
	Value	Mode		Description
	0	ONE		One channel.
	1	TWO		Two channels.
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	FORDER	0x0	RW	**Filter order**
	Configure order of the Filter.			
	Value	Mode		Description
	0	SECOND		Second order filter.
	1	THIRD		Third order filter.
	2	FOURTH		Fourth order filter.
	3	FIFTH		Fifth order filter.

20.6.7 PDM_CFG1 - PDM Core Configuration Register1

Offset	Bit Position																															
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset							0x0																0x0									
Access							RW																RW									
Name							DLYMUXSEL																PRESC									

Bit	Name	Reset	Access	Description
31:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:24	DLYMUXSEL	0x0	RW	**Data delay buffer mux selection**
	mux selection for delay buffer on data path			
23:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:0	PRESC	0x0	RW	**Prescalar Setting for PDM sample**
	Generate Decimation filter clock			

20.6.8 PDM_RXDATA - PDM Received Data Register

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R(r)																															
Name	RXDATA																															

Bit	Name	Reset	Access	Description
31:0	RXDATA	0x0	R(r)	**PDM received data**
	Use this register to access data from FIFO.			

20.6.9 PDM_IF - Interrupt Flag Register

Offset	Bit Position																															
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0	0x0	0x0
Access																													RW	RW	RW	RW
Name																													UF	OF	DVL	DV

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	UF	0x0	RW	**FIFO Undeflow Interrupt Flag**
	Set when the FIFO Underflow condition.			
2	OF	0x0	RW	**FIFO Overflow Interrupt Flag**
	Set when the FIFO Overflow condition.			
1	DVL	0x0	RW	**Data Valid Level Interrupt Flag**
	Set when the FIFO reaches to watermark level.			
0	DV	0x0	RW	**Data Valid Interrupt Flag**
	This interrupt is set after valid data available in FIFO.			

20.6.10 PDM_IEN - Interrupt Flag Register

Offset	Bit Position																															
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0	0x0	0x0
Access																													RW	RW	RW	RW
Name																													UF	OF	DVL	DV

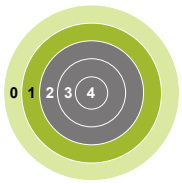
Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	UF	0x0	RW	**FIFO Undeflow Interrupt Enable**
	Set when the FIFO Underflow condition.			
2	OF	0x0	RW	**FIFO Overflow Interrupt Enable**
	Set when the FIFO Overflow condition.			
1	DVL	0x0	RW	**Data Valid Level Interrupt Enable**
	Set when the FIFO reaches to watermark level.			
0	DV	0x0	RW	**Data Valid Interrupt Enable**
	This interrupt is set after valid data available in FIFO.			

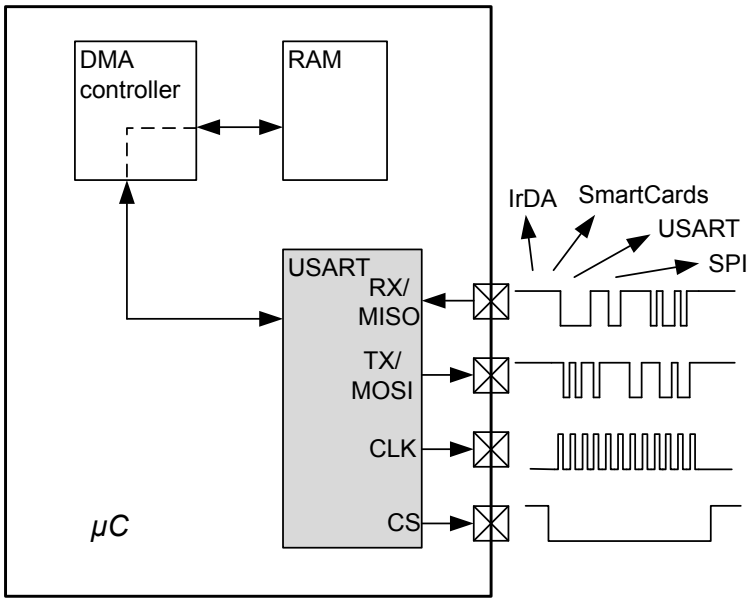
20.6.11 PDM_SYNCBUSY - Synchronization Busy Register

Offset	Bit Position																															
0x060	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			0x0
Access																													R			R
Name																													FIFOFLBUSY			SYNCBUSY

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	FIFOFLBUSY	0x0	R	**FIFO Flush Sync busy**
	Indicates FIFO Flush sync busy.			
2:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	SYNCBUSY	0x0	R	**sync busy**
	Indicates START sync busy.			

21. USART - Universal Synchronous Asynchronous Receiver/Transmitter





Quick Facts

What?

The USART handles high-speed UART, SPI-bus, SmartCards, and IrDA communication.

Why?

Serial communication is frequently used in embedded systems and the USART allows efficient communication with a wide range of external devices.

How?

The USART has a wide selection of operating modes, frame formats and baud rates. The multiprocessor mode allows the USART to remain idle when not addressed. Triple buffering and DMA support makes high data-rates possible with minimal CPU intervention and it is possible to transmit and receive large frames while the MCU remains in EM1 Sleep.

21.1 Introduction

The Universal Synchronous Asynchronous serial Receiver and Transmitter (USART) is a very flexible serial I/O module. It supports full duplex asynchronous UART communication as well as RS-485, SPI, MicroWire and 3-wire. It can also interface with ISO7816 SmartCards, and IrDA devices.

21.2 Features

- Asynchronous and synchronous (SPI) communication
- Full duplex and half duplex
- Separate TX/RX enable
- Separate receive / transmit multiple entry buffers, with additional separate shift registers
- Programmable baud rate, generated as an fractional division from the peripheral clock ($PCLK_{USARTn}$)
- Max bit-rate
  - SPI master mode, peripheral clock rate/2
  - SPI slave mode, peripheral clock rate/8
  - UART mode, peripheral clock rate/16, 8, 6, or 4
- Asynchronous mode supports
  - Majority vote baud-reception
  - False start-bit detection
  - Break generation/detection
  - Multi-processor mode
- Synchronous mode supports
  - All 4 SPI clock polarity/phase configurations
  - Master and slave mode
- Data can be transmitted LSB first or MSB first
- Configurable number of data bits, 4-16 (plus the parity bit, if enabled)
  - HW parity bit generation and check
- Configurable number of stop bits in asynchronous mode: 0.5, 1, 1.5, 2
- HW collision detection
- Multi-processor mode
- IrDA modulator
- SmartCard (ISO7816) mode
- I2S mode
- Separate interrupt vectors for receive and transmit interrupts
- Loopback mode
  - Half duplex communication
  - Communication debugging
- PRS RX input
- 8 bit Timer
- Hardware Flow Control
- Automatic Baud Rate Detection

21.3 Functional Description

An overview of the USART module is shown in Figure 21.1 USART Overview on page 558.

This section describes all posible USART features. Please refer to the Device Datasheet to see what features a specific USART instance supports.

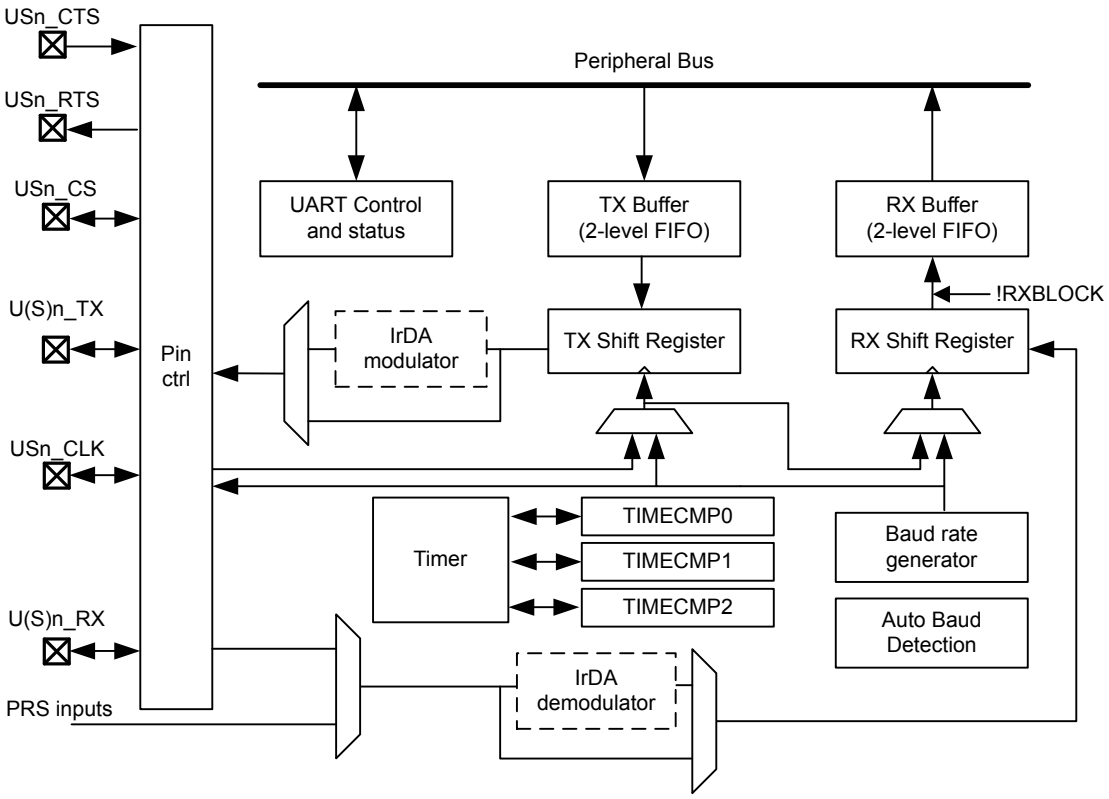


Figure 21.1. USART Overview

21.3.1 Modes of Operation

The USART operates in either asynchronous or synchronous mode.

In synchronous mode, a separate clock signal is transmitted with the data. This clock signal is generated by the bus master, and both the master and slave sample and transmit data according to this clock. Both master and slave modes are supported by the USART. The synchronous communication mode is compatible with the Serial Peripheral Interface Bus (SPI) standard.

In asynchronous mode, no separate clock signal is transmitted with the data on the bus. The USART receiver thus has to determine where to sample the data on the bus from the actual data. To make this possible, additional synchronization bits are added to the data when operating in asynchronous mode, resulting in a slight overhead.

Asynchronous or synchronous mode can be selected by configuring SYNC in USARTn_CTRL. The options are listed with supported protocols in Table 21.1 USART Asynchronous vs. Synchronous Mode on page 559. Full duplex and half duplex communication is supported in both asynchronous and synchronous mode.

Table 21.1. USART Asynchronous vs. Synchronous Mode

SYNC	Communication Mode	Supported Protocols
0	Asynchronous	RS-232, RS-485 (w/external driver), IrDA, ISO 7816
1	Synchronous	SPI, MicroWire, 3-wire

Table 21.2 USART Pin Usage on page 559 explains the functionality of the different USART pins when the USART operates in different modes. Pin functionality enclosed in square brackets is optional, and depends on additional configuration parameters. LOOPBK and MASTER are discussed in 21.3.2.14 Local Loopback and 21.3.3.3 Master Mode respectively.

Table 21.2. USART Pin Usage

SYNC	LOOPBK	MASTER	Pin functionality			
			U(S)n_TX (MOSI)	U(S)n_RX (MISO)	USn_CLK	USn_CS
0	0	x	Data out	Data in	-	[Driver enable]
0	1	x	Data out/in	-	-	[Driver enable]
1	0	0	Data in	Data out	Clock in	Slave select
1	0	1	Data out	Data in	Clock out	[Auto slave select]
1	1	0	Data out/in	-	Clock in	Slave select
1	1	1	Data out/in	-	Clock out	[Auto slave select]

21.3.2 Asynchronous Operation

21.3.2.1 Frame Format

The frame format used in asynchronous mode consists of a set of data bits in addition to bits for synchronization and optionally a parity bit for error checking. A frame starts with one start-bit (S), where the line is driven low for one bit-period. This signals the start of a frame, and is used for synchronization. Following the start bit are 4 to 16 data bits and an optional parity bit. Finally, a number of stop-bits, where the line is driven high, end the frame. An example frame is shown in Figure 21.2 USART Asynchronous Frame Format on page 560.

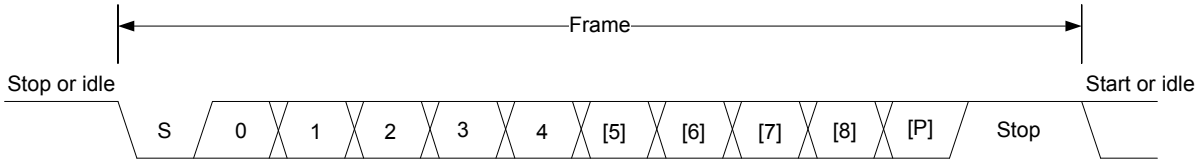


Figure 21.2. USART Asynchronous Frame Format

The number of data bits in a frame is set by DATABITS in USARTn_FRAME, see Table 21.3 USART Data Bits on page 560, and the number of stop-bits is set by STOPBITS in USARTn_FRAME, see Table 21.4 USART Stop Bits on page 560. Whether or not a parity bit should be included, and whether it should be even or odd is defined by PARITY, also in USARTn_FRAME. For communication to be possible, all parties of an asynchronous transfer must agree on the frame format being used.

Table 21.3. USART Data Bits

DATA BITS [3:0]	Number of Data bits
0001	4
0010	5
0011	6
0100	7
0101	8 (Default)
0110	9
0111	10
1000	11
1001	12
1010	13
1011	14
1100	15
1101	16

Table 21.4. USART Stop Bits

STOP BITS [1:0]	Number of Stop bits
00	0.5
01	1 (Default)
10	1.5
11	2

The order in which the data bits are transmitted and received is defined by MSBF in USARTn_CTRL. When MSBF is cleared, data in a frame is sent and received with the least significant bit first. When it is set, the most significant bit comes first.

The frame format used by the transmitter can be inverted by setting TXINV in USARTn_CTRL, and the format expected by the receiver can be inverted by setting RXINV in USARTn_CTRL. These bits affect the entire frame, not only the data bits. An inverted frame has a low idle state, a high start-bit, inverted data and parity bits, and low stop-bits.

21.3.2.2 Parity bit Calculation and Handling

When parity bits are enabled, hardware automatically calculates and inserts any parity bits into outgoing frames, and verifies the received parity bits in incoming frames. This is true for both asynchronous and synchronous modes, even though it is mostly used in asynchronous communication. The possible parity modes are defined in Table 21.5 USART Parity Bits on page 561. When even parity is chosen, a parity bit is inserted to make the number of high bits (data + parity) even. If odd parity is chosen, the parity bit makes the total number of high bits odd.

Table 21.5. USART Parity Bits

PARITY BITS [1:0]	Description
00	No parity bit (Default)
01	Reserved
10	Even parity
11	Odd parity

21.3.2.3 Clock Generation

The USART clock defines the transmission and reception data rate. When operating in asynchronous mode, the baud rate (bit-rate) is given by Figure 21.3 USART Baud Rate on page 562.

$$br = f_{PCLK}/(oversample \times (1 + USARTn\_CLKDIV/256))$$

Figure 21.3. USART Baud Rate

where f_{PCLK} is the peripheral clock ($PCLK_{USARTn}$) frequency and oversample is the oversampling rate as defined by OVS in USARTn_CTRL, see Table 21.6 USART Oversampling on page 562.

Table 21.6. USART Oversampling

OVS [1:0]	oversample
00	16
01	8
10	6
11	4

The USART has a fractional clock divider to allow the USART clock to be controlled more accurately than what is possible with a standard integral divider.

The clock divider used in the USART is a 20-bit value, with a 15-bit integral part and an 5-bit fractional part. The fractional part is configured in the lower 5 bits of DIV in USART_CLKDIV. The lowest achievable baud rate at 32 MHz is about 61 bauds/sec.

Fractional clock division is implemented by distributing the selected fraction over four baud periods. The fractional part of the divider tells how many of these periods should be extended by one peripheral clock cycle.

Given a desired baud rate brdesired, the clock divider USARTn_CLKDIV can be calculated by using Figure 21.4 USART Desired Baud Rate on page 562:

$$USARTn\_CLKDIV = 256 \times (f_{PCLK}/(oversample \times brdesired) - 1)$$

Figure 21.4. USART Desired Baud Rate

Table 21.7 USART Baud Rates @ 4MHz Peripheral Clock with 20 bit CLKDIV on page 562 shows a set of desired baud rates and how accurately the USART is able to generate these baud rates when running at a 4 MHz peripheral clock, using 16x or 8x oversampling.

Table 21.7. USART Baud Rates @ 4MHz Peripheral Clock with 20 bit CLKDIV

Desired baud rate [baud/s]	USARTn_OVS =00			USARTn_OVS =01		
	USARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %	USARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %
600	415,6563	600,015	0,003	832,3438	599,9925	-0,001
1200	207,3438	1199,94	-0,005	415,6563	1200,03	0,003
2400	103,1563	2400,24	0,010	207,3438	2399,88	-0,005
4800	51,09375	4799,04	-0,020	103,1563	4800,48	0,010
9600	25,03125	9603,842	0,040	51,09375	9598,08	-0,020
14400	16,375	14388,49	-0,080	33,71875	14401,44	0,010
19200	12,03125	19184,65	-0,080	25,03125	19207,68	0,040
28800	7,6875	28776,98	-0,080	16,375	28776,98	-0,080

Desired baud rate [baud/s]	USARTn_OVS =00			USARTn_OVS =01		
	USARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %	USARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %
38400	5,5	38461,54	0,160	12,03125	38369,3	-0,080
57600	3,34375	57553,96	-0,080	7,6875	57553,96	-0,080
76800	2,25	76923,08	0,160	5,5	76923,08	0,160
115200	1,15625	115942	0,644	3,34375	115107,9	-0,080
230400	0,09375	228571,4	-0,794	1,15625	231884,1	0,644

21.3.2.4 Auto Baud Detection

Setting AUTOBAUDEN in USARTn_CLKDIV uses the first frame received to automatically set the baud rate provided that it contains 0x55 (IrDA uses 0x00). AUTOBAUDEN can be used in a simple LIN configuration to auto detect the SYNC byte. The receiver will measure the number of local clock cycles between the beginning of the START bit and the beginning of the 8th data bit. The DIV field in USARTn_CLKDIV will be overwritten with the new value. The OVS in USARTn_CTRL and the +1 count of the Baud Rate equation are already factored into the result that gets written into the DIV field. To restart autobaud detection, clear AUTOBAUDEN and set it high again. Since the auto baud detection is done over 8 baud times, only the upper 3 bits of the fractional part of the clock divider are populated.

21.3.2.5 Data Transmission

Asynchronous data transmission is initiated by writing data to the transmit buffer using one of the methods described in 21.3.2.6 Transmit Buffer Operation. When the transmission shift register is empty and ready for new data, a frame from the transmit buffer is loaded into the shift register, and if the transmitter is enabled, transmission begins. When the frame has been transmitted, a new frame is loaded into the shift register if available, and transmission continues. If the transmit buffer is empty, the transmitter goes to an idle state, waiting for a new frame to become available.

Transmission is enabled through the command register USARTn_CMD by setting TXEN, and disabled by setting TXDIS in the same command register. When the transmitter is disabled using TXDIS, any ongoing transmission is aborted, and any frame currently being transmitted is discarded. When disabled, the TX output goes to an idle state, which by default is a high value. Whether or not the transmitter is enabled at a given time can be read from TXENS in USARTn_STATUS.

When the USART transmitter is enabled and there is no data in the transmit shift register or transmit buffer, the TXC flag in USARTn_STATUS and the TXC interrupt flag in USARTn_IF are set, signaling that the transmission is complete. The TXC status flag is cleared when a new frame becomes available for transmission, but the TXC interrupt flag must be cleared by software.

21.3.2.6 Transmit Buffer Operation

The transmit-buffer is a multiple entry FIFO buffer. A frame can be loaded into the buffer by writing to USARTn_TXDATA, USARTn_TXDATAX, USARTn_TXDOUBLE or USARTn_TXDOUBLEX. Using USARTn_TXDATA allows 8 bits to be written to the buffer, while using USARTn_TXDOUBLE will write 2 frames of 8 bits to the buffer. If 9-bit frames are used, the 9th bit of the frames will in these cases be set to the value of BIT8DV in USARTn_CTRL.

To set the 9th bit directly and/or use transmission control, USARTn_TXDATAX and USARTn_TXDOUBLEX must be used. USARTn_TXDATAX allows 9 data bits to be written, as well as a set of control bits regarding the transmission of the written frame. Every frame in the buffer is stored with 9 data bits and additional transmission control bits. USARTn_TXDOUBLEX allows two frames, complete with control bits to be written at once. When data is written to the transmit buffer using USARTn_TXDATAX and USARTn_TXDOUBLEX, the 9th bit(s) written to these registers override the value in BIT8DV in USARTn_CTRL, and alone define the 9th bits that are transmitted if 9-bit frames are used. Figure 21.5 USART Transmit Buffer Operation on page 564 shows the basics of the transmit buffer when DATABITS in USARTn_FRAME is configured to less than 10 bits.

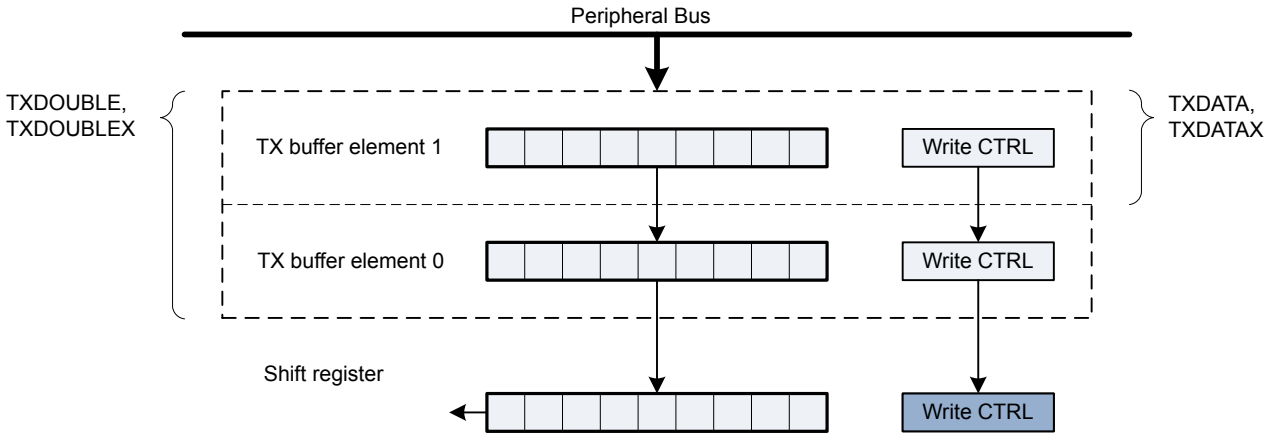


Figure 21.5. USART Transmit Buffer Operation

When writing more frames to the transmit buffer than there is free space for, the TXOF interrupt flag in USARTn_IF will be set, indicating the overflow. The data already in the transmit buffer is preserved in this case, and no data is written.

In addition to the interrupt flag TXC in USARTn_IF and status flag TXC in USARTn_STATUS which are set when the transmission is complete, TXBL in USARTn_STATUS and the TXBL interrupt flag in USARTn_IF are used to indicate the level of the transmit buffer. TXBIL in USARTn_CTRL controls the level at which these bits are set. If TXBIL is cleared, they are set whenever the transmit buffer becomes empty, and if TXBIL is set, they are set whenever the transmit buffer goes from full to half-full or empty. Both the TXBL status flag and the TXBL interrupt flag are cleared automatically when their condition becomes false.

The transmit buffer, including the transmit shift register can be cleared by setting CLEARTX in USARTn_CMD. This will prevent the USART from transmitting the data in the buffer and shift register, and will make them available for new data. Any frame currently being transmitted will not be aborted. Transmission of this frame will be completed.

21.3.2.7 Frame Transmission Control

The transmission control bits, which can be written using USARTn_TXDATAX and USARTn_TXDOUBLEX, affect the transmission of the written frame. The following options are available:

- Generate break: By setting TXBREAK, the output will be held low during the stop-bit period to generate a framing error. A receiver that supports break detection detects this state, allowing it to be used e.g. for framing of larger data packets. The line is driven high before the next frame is transmitted so the next start condition can be identified correctly by the recipient. Continuous breaks lasting longer than a USART frame are thus not supported by the USART. GPIO can be used for this.
- Disable transmitter after transmission: If TXDISAT is set, the transmitter is disabled after the frame has been fully transmitted.
- Enable receiver after transmission: If RXENAT is set, the receiver is enabled after the frame has been fully transmitted. It is enabled in time to detect a start-bit directly after the last stop-bit has been transmitted.
- Unblock receiver after transmission: If UBRXAT is set, the receiver is unblocked and RXBLOCK is cleared after the frame has been fully transmitted.
- Tristate transmitter after transmission: If TXTRIAT is set, TXTRI is set after the frame has been fully transmitted, tristating the transmitter output. Tristating of the output can also be performed automatically by setting AUTOTRI. If AUTOTRI is set TXTRI is always read as 0.

Note: When in SmartCard mode with repeat enabled, none of the actions, except generate break, will be performed until the frame is transmitted without failure. Generation of a break in SmartCard mode with repeat enabled will cause the USART to detect a NACK on every frame.

21.3.2.8 Data Reception

Data reception is enabled by setting RXEN in USARTn_CMD. When the receiver is enabled, it actively samples the input looking for a transition from high to low indicating the start baud of a new frame. When a start baud is found, reception of the new frame begins if the receive shift register is empty and ready for new data. When the frame has been received, it is pushed into the receive buffer, making the shift register ready for another frame of data, and the receiver starts looking for another start baud. If the receive buffer is full, the received frame remains in the shift register until more space in the receive buffer is available. If an incoming frame is detected while both the receive buffer and the receive shift register are full, the data in the shift register is overwritten, and the RXOF interrupt flag in USARTn_IF is set to indicate the buffer overflow.

The receiver can be disabled by setting the command bit RXDIS in USARTn_CMD. Any frame currently being received when the receiver is disabled is discarded. Whether or not the receiver is enabled at a given time can be read out from RXENS in USARTn_STATUS.

21.3.2.9 Receive Buffer Operation

When data becomes available in the receive buffer, the RXDATAV flag in USARTn_STATUS, and the RXDATAV interrupt flag in USARTn_IF are set, and when the buffer becomes full, RXFULL in USARTn_STATUS and the RXFULL interrupt flag in USARTn_IF are set. The status flags RXDATAV and RXFULL are automatically cleared by hardware when their condition is no longer true. This also goes for the RXDATAV interrupt flag, but the RXFULL interrupt flag must be cleared by software. When the RXFULL flag is set, notifying that the buffer is full, space is still available in the receive shift register for one more frame.

Data can be read from the receive buffer in a number of ways. USARTn_RXDATA gives access to the 8 least significant bits of the received frame, and USARTn_RXDOUBLE makes it possible to read the 8 least significant bits of two frames at once, pulling two frames from the buffer. To get access to the 9th, most significant bit, USARTn_RXDATAX must be used. This register also contains status information regarding the frame. USARTn_RXDOUBLEX can be used to get two frames complete with the 9th bits and status bits.

When a frame is read from the receive buffer using USARTn_RXDATA or USARTn_RXDATAX, the frame is pulled out of the buffer, making room for a new frame. USARTn_RXDOUBLE and USARTn_RXDOUBLEX pull two frames out of the buffer. If an attempt is done to read more frames from the buffer than what is available, the RXUF interrupt flag in USARTn_IF is set to signal the underflow, and the data read from the buffer is undefined.

Frames can be read from the receive buffer without removing the data by using USARTn_RXDATAXP and USARTn_RXDOUBLEXP. USARTn_RXDATAXP gives access the first frame in the buffer with status bits, while USARTn_RXDOUBLEXP gives access to both frames with status bits. The data read from these registers when the receive buffer is empty is undefined. If the receive buffer contains one valid frame, the first frame in USARTn_RXDOUBLEXP will be valid. No underflow interrupt is generated by a read using these registers, i.e. RXUF in USARTn_IF is never set as a result of reading from USARTn_RXDATAXP or USARTn_RXDOUBLEXP.

The basic operation of the receive buffer when DATABITS in USARTn_FRAME is configured to less than 10 bits is shown in Figure 21.6 USART Receive Buffer Operation on page 566.

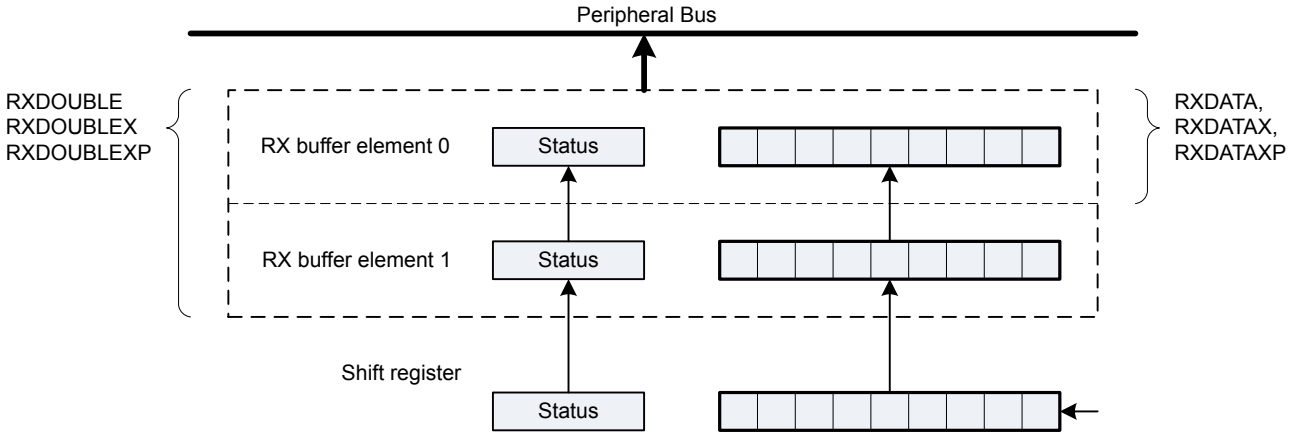


Figure 21.6. USART Receive Buffer Operation

The receive buffer, including the receive shift register can be cleared by setting CLEARRX in USARTn_CMD. Any frame currently being received will not be discarded.

21.3.2.10 Blocking Incoming Data

When using hardware frame recognition, as detailed in 21.3.2.20 Multi-Processor Mode and 21.3.2.21 Collision Detection, it is necessary to be able to let the receiver sample incoming frames without passing the frames to software by loading them into the receive buffer. This is accomplished by blocking incoming data.

Incoming data is blocked as long as RXBLOCK in USARTn_STATUS is set. When blocked, frames received by the receiver will not be loaded into the receive buffer, and software is not notified by the RXDATAV flag in USARTn_STATUS or the RXDATAV interrupt flag in USARTn_IF at their arrival. For data to be loaded into the receive buffer, RXBLOCK must be cleared in the instant a frame is fully received by the receiver. RXBLOCK is set by setting RXBLOCKEN in USARTn_CMD and disabled by setting RXBLOCKDIS also in USARTn_CMD. There is one exception where data is loaded into the receive buffer even when RXBLOCK is set. This is when an address frame is received when operating in multi-processor mode. See 21.3.2.20 Multi-Processor Mode for more information.

Frames received containing framing or parity errors will not result in the FERR and PERR interrupt flags in USARTn_IF being set while RXBLOCK in USARTn_STATUS is set. Hardware recognition is not applied to these erroneous frames, and they are silently discarded.

Note: If a frame is received while RXBLOCK in USARTn_STATUS is cleared, but stays in the receive shift register because the receive buffer is full, the received frame will be loaded into the receive buffer when space becomes available even if RXBLOCK is set at that time. The overflow interrupt flag RXOF in USARTn_IF will be set if a frame in the receive shift register, waiting to be loaded into the receive buffer is overwritten by an incoming frame even though RXBLOCK in USARTn_STATUS is set.

21.3.2.11 Clock Recovery and Filtering

The receiver samples the incoming signal at a rate 16, 8, 6 or 4 times higher than the given baud rate, depending on the oversampling mode given by OVS in USARTn_CTRL. Lower oversampling rates make higher baud rates possible, but give less room for errors.

When a high-to-low transition is registered on the input while the receiver is idle, this is recognized as a start-bit, and the baud rate generator is synchronized with the incoming frame.

For oversampling modes 16, 8 and 6, every bit in the incoming frame is sampled three times to gain a level of noise immunity. These samples are aimed at the middle of the bit-periods, as visualized in Figure 21.7 USART Sampling of Start and Data Bits on page 568. With OVS=0 in USARTn_CTRL, the start and data bits are thus sampled at locations 8, 9 and 10 in the figure, locations 4, 5 and 6 for OVS=1 and locations 3, 4, and 5 for OVS=2. The value of a sampled bit is determined by majority vote. If two or more of the three bit-samples are high, the resulting bit value is high. If the majority is low, the resulting bit value is low.

Majority vote is used for all oversampling modes except 4x oversampling. In this mode, a single sample is taken at position 3 as shown in Figure 21.7 USART Sampling of Start and Data Bits on page 568.

Majority vote can be disabled by setting MVDIS in USARTn_CTRL.

If the value of the start bit is found to be high, the reception of the frame is aborted, filtering out false start bits possibly generated by noise on the input.

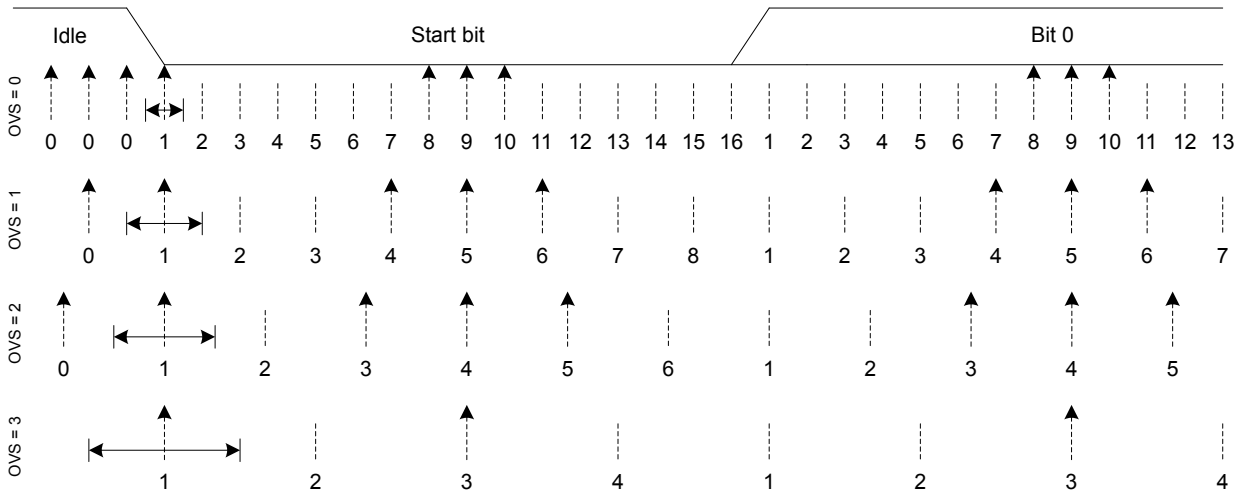


Figure 21.7. USART Sampling of Start and Data Bits

If the baud rate of the transmitter and receiver differ, the location each bit is sampled will be shifted towards the previous or next bit in the frame. This is acceptable for small errors in the baud rate, but for larger errors, it will result in transmission errors.

When the number of stop bits is 1 or more, stop bits are sampled like the start and data bits as seen in Figure 21.8 USART Sampling of Stop Bits when Number of Stop Bits are 1 or More on page 569. When a stop bit has been detected by sampling at positions 8, 9 and 10 for normal mode, or 4, 5 and 6 for smart mode, the USART is ready for a new start bit. As seen in Figure 21.8 USART Sampling of Stop Bits when Number of Stop Bits are 1 or More on page 569, a stop-bit of length 1 normally ends at c, but the next frame will be received correctly as long as the start-bit comes after position a for OVS=0 and OVS=3, and b for OVS=1 and OVS=2.

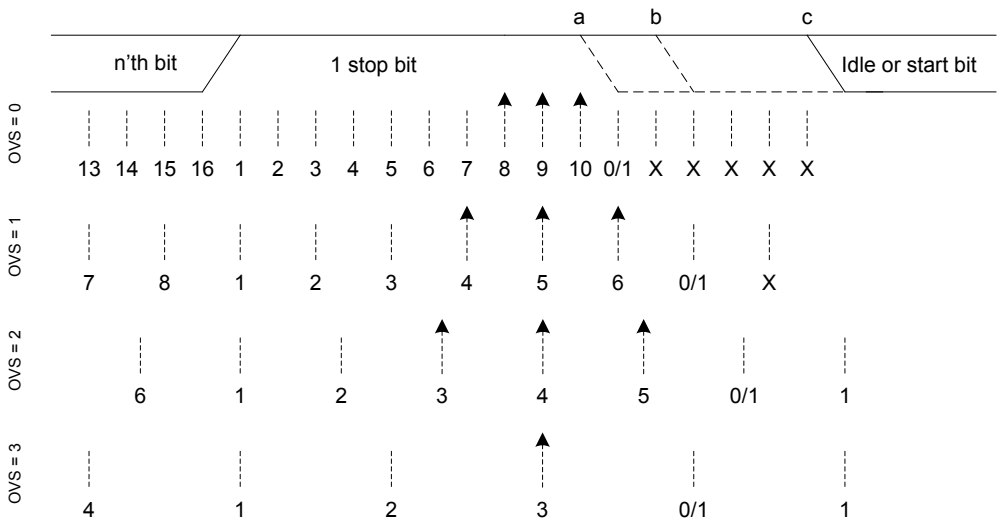


Figure 21.8. USART Sampling of Stop Bits when Number of Stop Bits are 1 or More

When working with stop bit lengths of half a baud period, the above sampling scheme no longer suffices. In this case, the stop-bit is not sampled, and no framing error is generated in the receiver if the stop-bit is not generated. The line must still be driven high before the next start bit however for the USART to successfully identify the start bit.

21.3.2.12 Parity Error

When parity bits are enabled, a parity check is automatically performed on incoming frames. When a parity error is detected in an incoming frame, the data parity error bit PERR in the frame is set, as well as the interrupt flag PERR in USARTn_IF. Frames with parity errors are loaded into the receive buffer like regular frames.

PERR can be accessed by reading the frame from the receive buffer using the USARTn_RXDATAX, USARTn_RXDATAXP, USARTn_RXDOUBLEX or USARTn_RXDOUBLEXP registers.

If ERRSTX in USARTn_CTRL is set, the transmitter is disabled on received parity and framing errors. If ERRSRX in USARTn_CTRL is set, the receiver is disabled on parity and framing errors.

21.3.2.13 Framing Error and Break Detection

A framing error is the result of an asynchronous frame where the stop bit was sampled to a value of 0. This can be the result of noise and baud rate errors, but can also be the result of a break generated by the transmitter on purpose.

When a framing error is detected in an incoming frame, the framing error bit FERR in the frame is set. The interrupt flag FERR in USARTn_IF is also set. Frames with framing errors are loaded into the receive buffer like regular frames.

FERR can be accessed by reading the frame from the receive buffer using the USARTn_RXDATAX, USARTn_RXDATAXP, USARTn_RXDOUBLEX or USARTn_RXDOUBLEXP registers.

If ERRSTX in USARTn_CTRL is set, the transmitter is disabled on parity and framing errors. If ERRSRX in USARTn_CTRL is set, the receiver is disabled on parity and framing errors.

21.3.2.14 Local Loopback

The USART receiver samples U(S)n_RX by default, and the transmitter drives U(S)n_TX by default. This is not the only option however. When LOOPBK in USARTn_CTRL is set, the receiver is connected to the U(S)n_TX pin as shown in Figure 21.9 USART Local Loopback on page 570. This is useful for debugging, as the USART can receive the data it transmits, but it is also used to allow the USART to read and write to the same pin, which is required for some half duplex communication modes. In this mode, the U(S)n_TX pin must be enabled as an output in the GPIO.

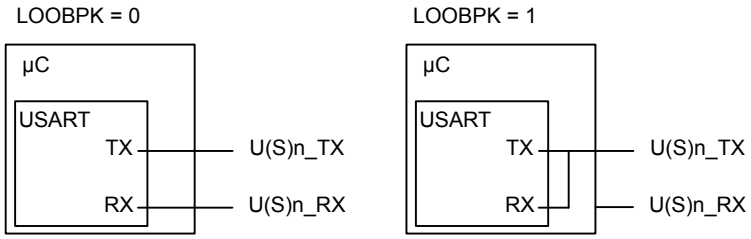


Figure 21.9. USART Local Loopback

21.3.2.15 Asynchronous Half Duplex Communication

When doing full duplex communication, two data links are provided, making it possible for data to be sent and received at the same time. In half duplex mode, data is only sent in one direction at a time. There are several possible half duplex setups, as described in the following sections.

21.3.2.16 Single Data-link

In this setup, the USART both receives and transmits data on the same pin. This is enabled by setting LOOPBK in USARTn_CTRL, which connects the receiver to the transmitter output. Because they are both connected to the same line, it is important that the USART transmitter does not drive the line when receiving data, as this would corrupt the data on the line.

When communicating over a single data-link, the transmitter must thus be tristated whenever not transmitting data. This is done by setting the command bit TXTRIEN in USARTn_CMD, which tristates the transmitter. Before transmitting data, the command bit TXTRIDIS, also in USARTn_CMD, must be set to enable transmitter output again. Whether or not the output is tristated at a given time can be read from TXTRI in USARTn_STATUS. If TXTRI is set when transmitting data, the data is shifted out of the shift register, but is not put out on U(S)n_TX.

When operating a half duplex data bus, it is common to have a bus master, which first transmits a request to one of the bus slaves, then receives a reply. In this case, the frame transmission control bits, which can be set by writing to USARTn_TXDATAX, can be used to make the USART automatically disable transmission, tristate the transmitter and enable reception when the request has been transmitted, making it ready to receive a response from the slave.

The timer, 21.3.10 Timer, can also be used to add delay between the RX and TX frames so that the interrupt service routine has time to process data that was just received before transmitting more data. Also hardware flow control is another method to insert time for processing the frame. RTS and CTS can be used to halt either the link partner's transmitter or the local transmitter. See the section on hardware flow control, 21.3.4 Hardware Flow Control, for more details.

Tristating the transmitter can also be performed automatically by the USART by using AUTOTRI in USARTn_CTRL. When AUTOTRI is set, the USART automatically tristates U(S)n_TX whenever the transmitter is idle, and enables transmitter output when the transmitter goes active. If AUTOTRI is set TXTRI is always read as 0.

Note: Another way to tristate the transmitter is to enable wired-and or wired-or mode in GPIO. For wired-and mode, outputting a 1 will be the same as tristating the output, and for wired-or mode, outputting a 0 will be the same as tristating the output. This can only be done on buses with a pull-up or pull-down resistor respectively.

21.3.2.17 Single Data-link with External Driver

Some communication schemes, such as RS-485 rely on an external driver. Here, the driver has an extra input which enables it, and instead of tristating the transmitter when receiving data, the external driver must be disabled.

This can be done manually by assigning a GPIO to turn the driver on or off, or it can be handled automatically by the USART. If AUTOCS in USARTn_CTRL is set, the USn_CS output is automatically activated a configurable number of baud periods before the transmitter starts transmitting data, and deactivated a configurable number of baud periods after the last bit has been transmitted and there is no more data in the transmit buffer to transmit. The number of baud periods are controlled by CSSETUP and CSHOLD in USARTn_TIMING. This feature can be used to turn the external driver on when transmitting data, and turn it off when the data has been transmitted.

The timer, 21.3.10 Timer, can also be used to configure CSSETUP and CSHOLD values between 1 to 256 baud-times by using TCMPVAL0, TCMPVAL1, or TCMPVAL2 for the TX sequencer.

USn_CS is immediately deasserted when the transmitter becomes disabled.

Note: When using CSSETUP in asynchronous mode with AUTOCS (USARTn_CTRL.SYNC = 0, USARTn_CTRL.AUTOCS = 1), TXDELAY in USARTn_TIMING should be set to 1.

Figure 21.10 USART Half Duplex Communication with External Driver on page 571 shows an example configuration where USn_CS is used to automatically enable and disable an external driver.

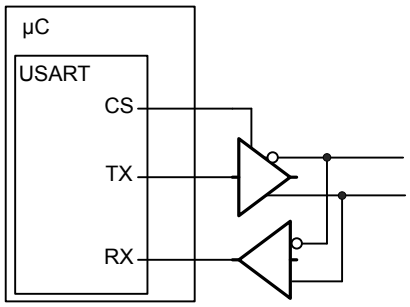


Figure 21.10. USART Half Duplex Communication with External Driver

The USn_CS output is active low by default, but its polarity can be changed with CSINV in USARTn_CTRL. AUTOCS works regardless of which mode the USART is in, so this functionality can also be used for automatic chip/slave select when in synchronous mode (e.g. SPI).

21.3.2.18 Two Data-links

Some limited devices only support half duplex communication even though two data links are available. In this case software is responsible for making sure data is not transmitted when incoming data is expected.

TXARXnEN in USARTn_TRIGCTRL may be used to automatically start transmission after the end of the RX frame plus any TXSTDELAY and CSSETUP delay in USARTn_TIMING. For enabling the receiver either use RXENAT in USARTn_TXDATAX or RXATXnEN in USARTn_TRIGCTRL.

21.3.2.19 Large Frames

As each frame in the transmit and receive buffers holds a maximum of 9 bits, both the elements in the buffers are combined when working with USART-frames of 10 or more data bits.

To transmit such a frame, at least two elements must be available in the transmit buffer. If only one element is available, the USART will wait for the second element before transmitting the combined frame. Both the elements making up the frame are consumed when transmitting such a frame.

When using large frames, the 9th bits in the buffers are unused. For an 11 bit frame, the 8 least significant bits are thus taken from the first element in the buffer, and the 3 remaining bits are taken from the second element as shown in Figure 21.11 USART Transmission of Large Frames on page 572. The first element in the transmit buffer, i.e. element 0 in Figure 21.11 USART Transmission of Large Frames on page 572 is the first element written to the FIFO, or the least significant byte when writing two bytes at a time using USARTn_TXDOUBLE.

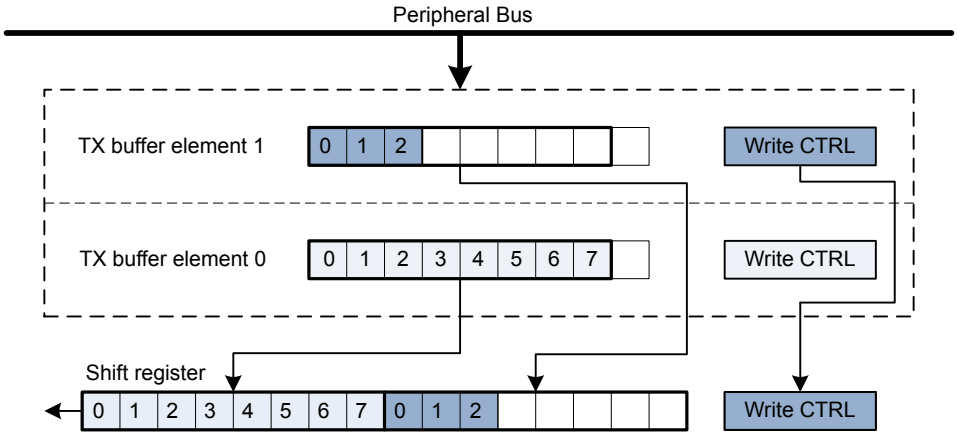


Figure 21.11. USART Transmission of Large Frames

As shown in Figure 21.11 USART Transmission of Large Frames on page 572, frame transmission control bits are taken from the second element in FIFO.

The two buffer elements can be written at the same time using the USARTn_TXDOUBLE or USARTn_TXDOUBLEX register. The TXDATAX0 bitfield then refers to buffer element 0, and TXDATAX1 refers to buffer element 1.

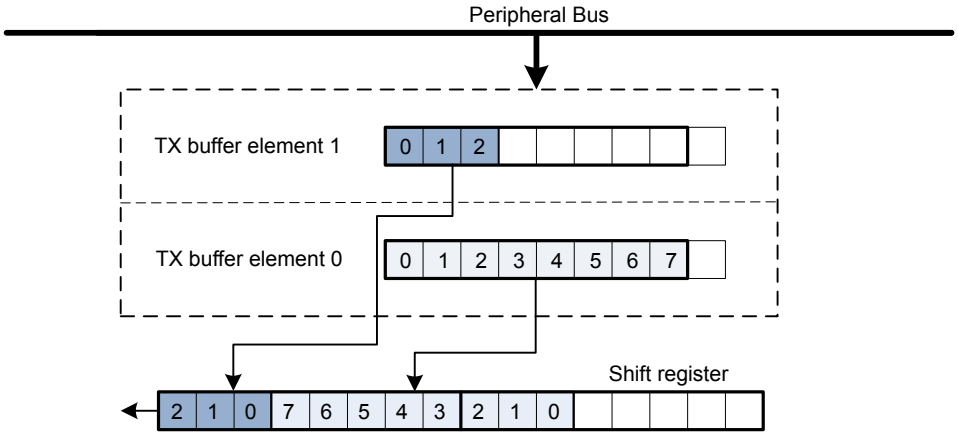


Figure 21.12. USART Transmission of Large Frames, MSBF

Figure 21.12 USART Transmission of Large Frames, MSBF on page 572 illustrates the order of the transmitted bits when an 11 bit frame is transmitted with MSBF set. If MSBF is set and the frame is smaller than 10 bits, only the contents of transmit buffer 0 will be transmitted.

When receiving a large frame, BYTESWAP in USARTn_CTRL determines the order the way the large frame is split into the two buffer elements. If BYTESWAP is cleared, the least significant 8 bits of the received frame are loaded into the first element of the receive buffer, and the remaining bits are loaded into the second element, as shown in Figure 21.13 USART Reception of Large Frames on page 573. The first byte read from the buffer thus contains the 8 least significant bits. Set BYTESWAP to reverse the order.

The status bits are loaded into both elements of the receive buffer. The frame is not moved from the receive shift register before there are two free spaces in the receive buffer.

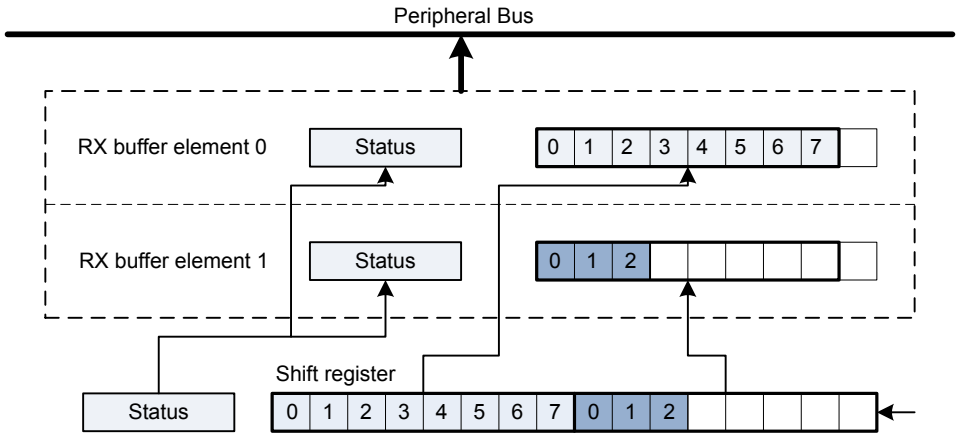


Figure 21.13. USART Reception of Large Frames

The two buffer elements can be read at the same time using the USARTn_RXDOUBLE or USARTn_RXDOUBLEX register. RXDATA0 then refers to buffer element 0 and RXDATA1 refers to buffer element 1.

Large frames can be used in both asynchronous and synchronous modes.

21.3.2.20 Multi-Processor Mode

To simplify communication between multiple processors, the USART supports a special multi-processor mode. In this mode the 9th data bit in each frame is used to indicate whether the content of the remaining 8 bits is data or an address.

When multi-processor mode is enabled, an incoming 9-bit frame with the 9th bit equal to the value of MPAB in USARTn_CTRL is identified as an address frame. When an address frame is detected, the MPAF interrupt flag in USARTn_IF is set, and the address frame is loaded into the receive register. This happens regardless of the value of RXBLOCK in USARTn_STATUS.

Multi-processor mode is enabled by setting MPM in USARTn_CTRL, and the value of the 9th bit in address frames can be set in MPAB. Note that the receiver must be enabled for address frames to be detected. The receiver can be blocked however, preventing data from being loaded into the receive buffer while looking for address frames.

When a slave has received an address frame and wants to receive the following data, it must make sure the receiver is unblocked before the next frame has been completely received in order to prevent data loss.

BIT8DV in USARTn_CTRL can be used to specify the value of the 9th bit without writing to the transmit buffer with USARTn_TXDATAX or USARTn_TXDOUBLEX, giving higher efficiency in multi-processor mode, as the 9th bit is only set when writing address frames, and 8-bit writes to the USART can be used when writing the data frames.

21.3.2.21 Collision Detection

The USART supports a basic form of collision detection. When the receiver is connected to the output of the transmitter, either by using the LOOPBK bit in USARTn_CTRL or through an external connection, this feature can be used to detect whether data transmitted on the bus by the USART did get corrupted by a simultaneous transmission by another device on the bus.

For collision detection to be enabled, CCEN in USARTn_CTRL must be set, and the receiver enabled. The data sampled by the receiver is then continuously compared with the data output by the transmitter. If they differ, the CCF interrupt flag in USARTn_IF is set. The collision check includes all bits of the transmitted frames. The CCF interrupt flag is set once for each bit sampled by the receiver that differs from the bit output by the transmitter. When the transmitter output is disabled, i.e. the transmitter is tristated, collisions are not registered.

21.3.2.22 SmartCard Mode

In SmartCard mode, the USART supports the ISO 7816 I/O line T0 mode. With exception of the stop-bits (guard time), the 7816 data frame is equal to the regular asynchronous frame. In this mode, the receiver pulls the line low for one baud, half a baud into the guard time to indicate a parity error. This NAK can for instance be used by the transmitter to re-transmit the frame. SmartCard mode is a half duplex asynchronous mode, so the transmitter must be tristated whenever not transmitting data.

To enable SmartCard mode, set SCMODE in USARTn_CTRL, set the number of databits in a frame to 8, and configure the number of stopbits to 1.5 by writing to STOPBITS in USARTn_FRAME.

The SmartCard mode relies on half duplex communication on a single line, so for it to work, both the receiver and transmitter must work on the same line. This can be achieved by setting LOOPBK in USARTn_CTRL or through an external connection. The TX output should be configured as open-drain in the GPIO module.

When no parity error is identified by the receiver, the data frame is as shown in Figure 21.14 USART ISO 7816 Data Frame Without Error on page 574. The frame consists of 8 data bits, a parity bit, and 2 stop bits. The transmitter does not drive the output line during the guard time.

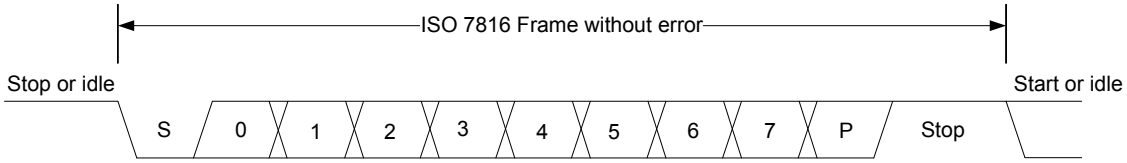


Figure 21.14. USART ISO 7816 Data Frame Without Error

If a parity error is detected by the receiver, it pulls the line I/O line low after half a stop bit, see Figure 21.15 USART ISO 7816 Data Frame With Error on page 574. It holds the line low for one bit-period before it releases the line. In this case, the guard time is extended by one bit period before a new transmission can start, resulting in a total of 3 stop bits.

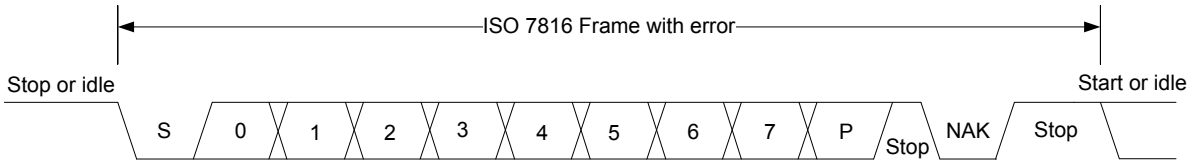


Figure 21.15. USART ISO 7816 Data Frame With Error

On a parity error, the NAK is generated by hardware. The NAK generated by the receiver is sampled as the stop-bit of the frame. Because of this, parity errors when in SmartCard mode are reported with both a parity error and a framing error.

When transmitting a T0 frame, the USART receiver on the transmitting side samples position 16, 17 and 18 in the stop-bit to detect the error signal when in 16x oversampling mode as shown in Figure 21.16 USART SmartCard Stop Bit Sampling on page 575. Sampling at this location places the stop-bit sample in the middle of the bit-period used for the error signal (NAK).

If a NAK is transmitted by the receiver, it will thus appear as a framing error at the transmitter, and the FERR interrupt flag in USARTn_IF will be set. If SCRETRANS USARTn_CTRL is set, the transmitter will automatically retransmit a NACK'ed frame. The transmitter will retransmit the frame until it is ACK'ed by the receiver. This only works when the number of databits in a frame is configured to 8.

Set SKIPPERRF in USARTn_CTRL to make the receiver discard frames with parity errors. The PERR interrupt flag in USARTn_IF is set when a frame is discarded because of a parity error.

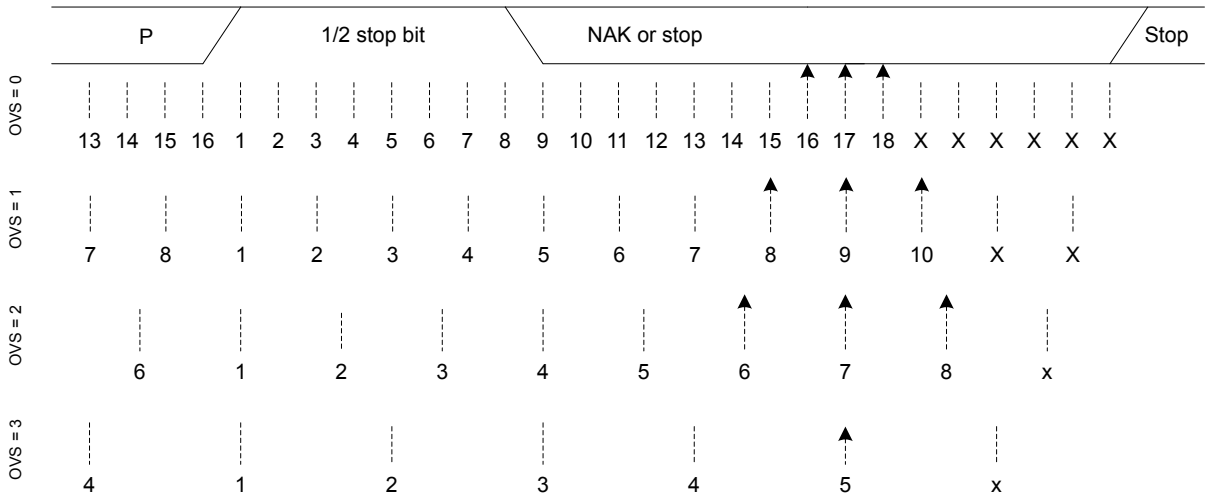


Figure 21.16. USART SmartCard Stop Bit Sampling

For communication with a SmartCard, a clock signal needs to be generated for the card. This clock output can be generated using one of the timers. See the ISO 7816 specification for more info on this clock signal.

SmartCard T1 mode is also supported. The T1 frame format used is the same as the asynchronous frame format with parity bit enabled and one stop bit. The USART must then be configured to operate in asynchronous half duplex mode.

21.3.3 Synchronous Operation

Most of the features in asynchronous mode are available in synchronous mode. Multi-processor mode can be enabled for 9-bit frames, loopback is available and collision detection can be performed.

21.3.3.1 Frame Format

The frames used in synchronous mode need no start and stop bits since a single clock is available to all parts participating in the communication. Parity bits cannot be used in synchronous mode.

The USART supports frame lengths of 4 to 16 bits per frame. Larger frames can be simulated by transmitting multiple smaller frames, i.e. a 22 bit frame can be sent using two 11-bit frames, and a 21 bit frame can be generated by transmitting three 7-bit frames. The number of bits in a frame is set using DATABITS in USARTn_FRAME.

The frames in synchronous mode are by default transmitted with the least significant bit first like in asynchronous mode. The bit-order can be reversed by setting MSBF in USARTn_CTRL.

The frame format used by the transmitter can be inverted by setting TXINV in USARTn_CTRL, and the format expected by the receiver can be inverted by setting RXINV, also in USARTn_CTRL.

21.3.3.2 Clock Generation

The bit-rate in synchronous mode is given by Figure 21.17 USART Synchronous Mode Bit Rate on page 576. As in the case of asynchronous operation, the clock division factor have a 15-bit integral part and a 5-bit fractional part.

$$br = f_{PCLK}/(2 \times (1 + USARTn\_CLKDIV/256))$$

Figure 21.17. USART Synchronous Mode Bit Rate

Given a desired baud rate brdesired, the clock divider USARTn_CLKDIV can be calculated using Figure 21.18 USART Synchronous Mode Clock Division Factor on page 576

$$USARTn\_CLKDIV = 256 \times (f_{PCLK}/(2 \times brdesired) - 1)$$

Figure 21.18. USART Synchronous Mode Clock Division Factor

When the USART operates in master mode, the highest possible bit rate is half the peripheral clock rate. When operating in slave mode however, the highest bit rate is an eighth of the peripheral clock:

- Master mode: $br_{max} = f_{PCLK}/2$
- Slave mode: $br_{max} = f_{PCLK}/8$

On every clock edge data on the data lines, MOSI and MISO, is either set up or sampled. When CLKPHA in USARTn_CTRL is cleared, data is sampled on the leading clock edge and set-up is done on the trailing edge. If CLKPHA is set however, data is set-up on the leading clock edge, and sampled on the trailing edge. In addition to this, the polarity of the clock signal can be changed by setting CLKPOL in USARTn_CTRL, which also defines the idle state of the clock. This results in four different modes which are summarized in Table 21.8 USART SPI Modes on page 576. Figure 21.19 USART SPI Timing on page 576 shows the resulting timing of data set-up and sampling relative to the bus clock.

Table 21.8. USART SPI Modes

SPI mode	CLKPOL	CLKPHA	Leading edge	Trailing edge
0	0	0	Rising, sample	Falling, set-up
1	0	1	Rising, set-up	Falling, sample
2	1	0	Falling, sample	Rising, set-up
3	1	1	Falling, set-up	Rising, sample

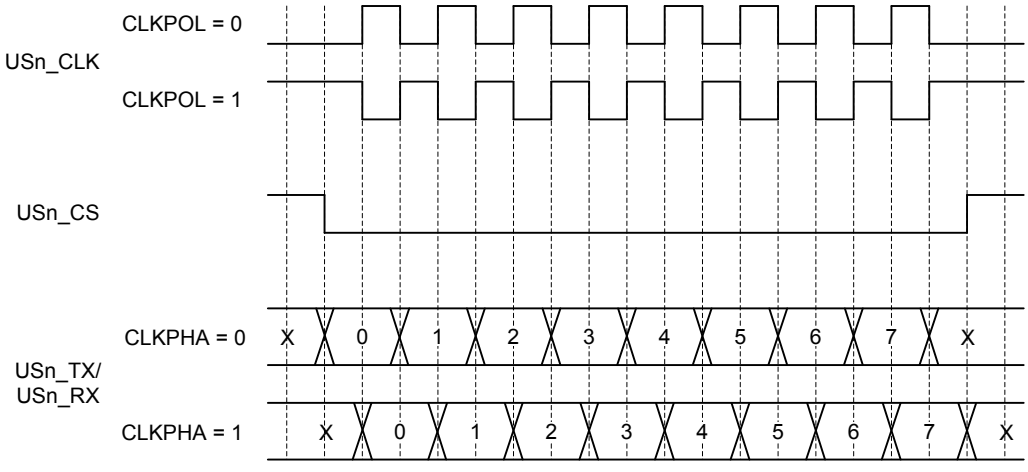


Figure 21.19. USART SPI Timing

If CPHA=1, the TX underflow flag, TXUF, will be set on the first setup clock edge of a frame in slave mode if TX data is not available. If CPHA=0, TXUF is set if data is not available in the transmit buffer three PCLK cycles prior to the first sample clock edge. The RXDATAV flag is updated on the last sample clock edge of a transfer, while the RX overflow interrupt flag, RXOF, is set on the first sample

clock edge if the receive buffer overflows. When a transfer has been performed, interrupt flags TXBL and TXC are updated on the first setup clock edge of the succeeding frame, or when CS is deasserted.

21.3.3.3 Master Mode

When in master mode, the USART is in full control of the data flow on the synchronous bus. When operating in full duplex mode, the slave cannot transmit data to the master without the master transmitting to the slave. The master outputs the bus clock on USn_CLK.

Communication starts whenever there is data in the transmit buffer and the transmitter is enabled. The USART clock then starts, and the master shifts bits out from the transmit shift register using the internal clock.

When there are no more frames in the transmit buffer and the transmit shift register is empty, the clock stops, and communication ends. When the receiver is enabled, it samples data using the internal clock when the transmitter transmits data. Operation of the RX and TX buffers is as in asynchronous mode.

21.3.3.4 Operation of USn_CS Pin

When operating in master mode, the USn_CS pin can have one of two functions, or it can be disabled.

If USn_CS is configured as an output, it can be used to automatically generate a chip select for a slave by setting AUTOCS in USARTn_CTRL. If AUTOCS is set, USn_CS is activated before a transmission begins, and deactivated after the last bit has been transmitted and there is no more data in the transmit buffer.

The time between when CS is asserted and the first bit is transmitted can be controlled using the USART Timer and with CSSETUP in USARTn_TIMING. Any of the three comparators can be used to set this delay. If new data is ready for transmission before CS is deasserted, the data is sent without deasserting CS in between. CSHOLD in USARTn_TIMING keeps CS asserted after the end of frame for the number of baud-times specified.

By default, USn_CS is active low, but its polarity can be inverted by setting CSINV in USARTn_CTRL.

When USn_CS is configured as an input, it can be used by another master that wants control of the bus to make the USART release it. When USn_CS is driven low, or high if CSINV is set, the interrupt flag SSM in USARTn_IF is set, and if CSMA in USARTn_CTRL is set, the USART goes to slave mode.

21.3.3.5 AUTOTX

A synchronous master is required to transmit data to a slave in order to receive data from the slave. In some cases, only a few words are transmitted and a lot of data is then received from the slave. In that case, one solution is to keep feeding the TX with data to transmit, but that consumes system bandwidth. Instead AUTOTX can be used.

When AUTOTX in USARTn_CTRL is set, the USART transmits data as long as there is available space in the RX shift register for the chosen frame size. This happens even though there is no data in the TX buffer. The TX underflow interrupt flag TXUF in USARTn_IF is set on the first word that is transmitted which does not contain valid data.

During AUTOTX the USART will always send the previous sent bit, thus reducing the number of transitions on the TX output. So if the last bit sent was a 0, 0's will be sent during AUTOTX and if the last bit sent was a 1, 1's will be sent during AUTOTX.

21.3.3.6 Slave Mode

When the USART is in slave mode, data transmission is not controlled by the USART, but by an external master. The USART is therefore not able to initiate a transmission, and has no control over the number of bytes written to the master.

The output and input to the USART are also swapped when in slave mode, making the receiver take its input from USn_TX (MOSI) and the transmitter drive USn_RX (MISO).

To transmit data when in slave mode, the slave must load data into the transmit buffer and enable the transmitter. The data will remain in the USART until the master starts a transmission by pulling the USn_CS input of the slave low and transmitting data. For every frame the master transmits to the slave, a frame is transferred from the slave to the master. After a transmission, MISO remains in the same state as the last bit transmitted. This also applies if the master transmits to the slave and the slave TX buffer is empty.

If the transmitter is enabled in synchronous slave mode and the master starts transmission of a frame, the underflow interrupt flag TXUF in USARTn_IF will be set if no data is available for transmission to the master.

If the slave needs to control its own chip select signal, this can be achieved by clearing CSPEN in the GPIO_USARTn_ROUTEEN register. The internal chip select signal can then be controlled through CSINV in the CTRL register. The chip select signal will be CSINV inverted, i.e. if CSINV is cleared, the chip select is active and vice versa.

21.3.3.7 Synchronous Half Duplex Communication

Half duplex communication in synchronous mode is very similar to half duplex communication in asynchronous mode as detailed in 21.3.2.15 Asynchronous Half Duplex Communication. The main difference is that in this mode, the master must generate the bus clock even when it is not transmitting data, i.e. it must provide the slave with a clock to receive data. To generate the bus clock, the master should transmit data with the transmitter tristated, i.e. TXTRI in USARTn_STATUS set, when receiving data. If 2 bytes are expected from the slave, then transmit 2 bytes with the transmitter tristated, and the slave uses the generated bus clock to transmit data to the master. TXTRI can be set by setting the TXTRIEN command bit in USARTn_CMD.

Note: When operating as SPI slave in half duplex mode, TX has to be tristated (not disabled) during data reception if the slave is to transmit data in the current transfer.

21.3.3.8 I2S

I2S is a synchronous format for transmission of audio data. The frame format is 32-bit, but since data is always transmitted with MSB first, an I2S device operating with 16-bit audio may choose to only process the 16 msb of the frame, and only transmit data in the 16 msb of the frame.

In addition to the bit clock used for regular synchronous transfers, I2S mode uses a separate word clock. When operating in mono mode, with only one channel of data, the word clock pulses once at the start of each new word. In stereo mode, the word clock toggles at the start of new words, and also gives away whether the transmitted word is for the left or right audio channel; A word transmitted while the word clock is low is for the left channel, and a word transmitted while the word clock is high is for the right.

When operating in I2S mode, the CS pin is used as a the word clock. In master mode, this is automatically driven by the USART, and in slave mode, the word clock is expected from an external master.

21.3.3.9 Word Format

The general I2S word format is 32 bits wide, but the USART also supports 16-bit and 8-bit words. In addition to this, it can be specified how many bits of the word should actually be used by the USART. These parameters are given by FORMAT in USARTn_I2SCTRL.

As an example, configuring FORMAT to using a 32-bit word with 16-bit data will make each word on the I2S bus 32-bits wide, but when receiving data through the USART, only the 16 most significant bits of each word can be read out of the USART. Similarly, only the 16 most significant bits have to be written to the USART when transmitting. The rest of the bits will be transmitted as zeroes.

21.3.3.10 Major Modes

The USART supports a set of different I2S formats as shown in Table 21.9 USART I2S Modes on page 579, but it is not limited to these modes. MONO, JUSTIFY and DELAY in USARTn_I2SCTRL can be mixed and matched to create an appropriate format. MONO enables mono mode, i.e. one data stream instead of two which is the default. JUSTIFY aligns data within a word on the I2S bus, either left or right which can bee seen in figures Figure 21.22 USART Left-justified I2S waveform on page 580 and Figure 21.23 USART Right-justified I2S waveform on page 580. Finally, DELAY specifies whether a new I2S word should be started directly on the edge of the word-select signal, or one bit-period after the edge.

Table 21.9. USART I2S Modes

Mode	MONO	JUSTIFY	DELAY	CLKPOL
Regular I2S	0	0	1	0
Left-Justified	0	0	0	1
Right-Justified	0	1	0	1
Mono	1	0	0	0

The regular I2S waveform is shown in Figure 21.20 USART Standard I2S waveform on page 579 and Figure 21.21 USART Standard I2S waveform (reduced accuracy) on page 579. The first figure shows a waveform transmitted with full accuracy. The wordlength can be configured to 32-bit, 16-bit or 8-bit using FORMAT in USARTn_I2SCTRL. In the second figure, I2S data is transmitted with reduced accuracy, i.e. the data transmitted has less bits than what is possible in the bus format.

Note that the msb of a word transmitted in regular I2S mode is delayed by one cycle with respect to word select

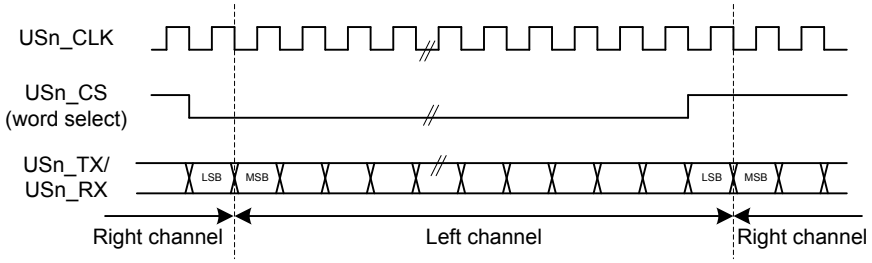


Figure 21.20. USART Standard I2S waveform

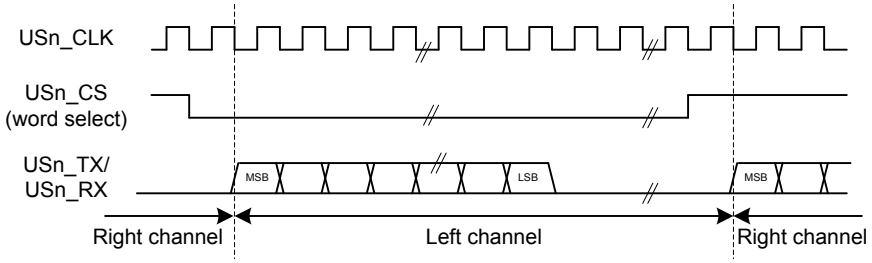


Figure 21.21. USART Standard I2S waveform (reduced accuracy)

A left-justified stream is shown in Figure 21.22 USART Left-justified I2S waveform on page 580. Note that the MSB comes directly after the edge on the word-select signal in contradiction to the regular I2S waveform where it comes one bit-period after.

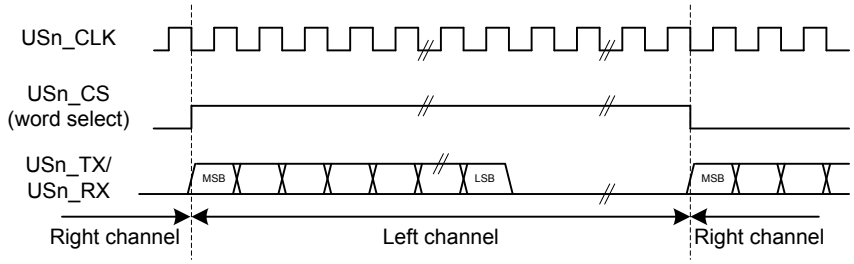


Figure 21.22. USART Left-justified I2S waveform

A right-justified stream is shown in Figure 21.23 USART Right-justified I2S waveform on page 580. The left and right justified streams are equal when the data-size is equal to the word-width.

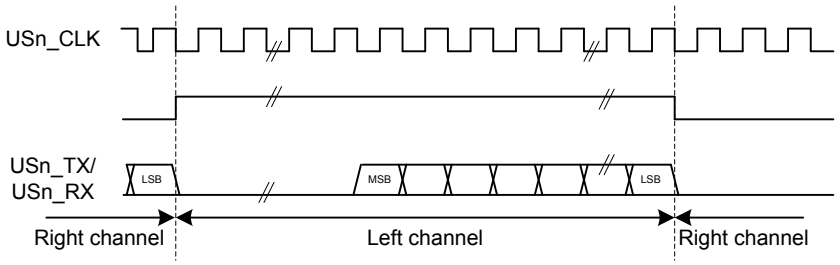


Figure 21.23. USART Right-justified I2S waveform

In mono-mode, the word-select signal pulses at the beginning of each word instead of toggling for each word. Mono I2S waveform is shown in Figure 21.24 USART Mono I2S waveform on page 580.

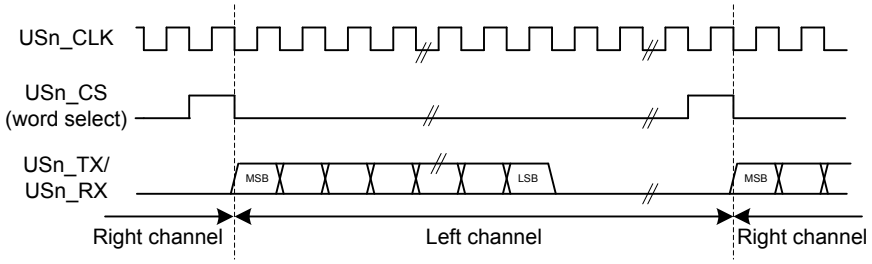


Figure 21.24. USART Mono I2S waveform

21.3.3.11 Using I2S Mode

When using the USART in I2S mode, DATABITS in USARTn_FRAME must be set to 8 or 16 data-bits. 8 databits can be used in all modes, and 16 can be used in the modes where the number of bytes in the I2S word is even. In addition to this, MSBF in USARTn_CTRL should be set, and CLKPOL and CLKPHA in USARTn_CTRL should be cleared.

The USART does not have separate TX and RX buffers for left and right data, so when using I2S in stereo mode, the application must keep track of whether the buffers contain left or right data. This can be done by observing TXBLRIGHT, RXDATAVRIGHT and RXFULLRIGHT in USARTn_STATUS. TXBLRIGHT tells whether TX is expecting data for the left or right channel. It will be set with TXBL if right data is expected. The receiver will set RXDATAVRIGHT if there is at least one right element in the buffer, and RXFULLRIGHT if the buffer is full of right elements.

When using I2S with DMA, separate DMA requests can be used for left and right data by setting DMASPLIT in USARTn_I2SCTRL.

In both master and slave mode the USART always starts transmitting on the LEFT channel after being enabled. In master mode, the transmission will stop if TX becomes empty. In that case, TXC is set. Continuing the transmission in this case will make the data-stream continue where it left off. To make the USART start on the LEFT channel after going empty, disable and re-enable TX.

21.3.4 Hardware Flow Control

Hardware flow control can be used to hold off the link partner's transmission until RX buffer space is available. The RTS and CTS signals are enabled and configured using the GPIO_DBUSUSARTn_ROUTEEN, GPIO_DBUSUSARTn_RTSROUTEx and GPIO_DBUSUSARTn_CTSROUTE registers. RTS is an out going signal which indicates that RX buffer space is available to receive a frame. The link partner is being requested to send its data when RTS is asserted. CTS is an incoming signal to stop the next TX data from going out. When CTS is negated, the frame currently being transmitted is completed before stopping. CTS indicates that the link partner has RX buffer space available, and the local transmitter is clear to send. Also use CTSEN in USARTn_CTLX to enable the CTS input into the TX sequencer. For debug use set DBGHALT in USARTn_CTRLX which will force the RTS to request one frame from the link partner when the CPU core single steps.

21.3.5 Debug Halt

When DBGHALT in USART_CTRLX is clear, RTS is only dependent on the RX buffer having space available to receive data. Incoming data is always received until both the RX buffer is full and the RX shift register is full regardless of the state of DBGHALT or chip halt. Additional incoming data is discarded. When DBGHALT is set, RTS deasserts on RX buffer full or when chip halt is high. However, a low pulse detected on chip halt will keep RTS asserted when no frame is being received. At the start of frame reception, RTS will deassert if chip halt is high and DBGHALT is set. This behavior allows single stepping to pulse the chip halt low for a cycle, and receive the next frame. The link partner must stop transmitting when RTS is deasserted, or the RX buffer could overflow. All data in the transmit buffer is sent out even when chip halt is asserted; therefore, the DMA will need to be set to stop sending the USART TX data during chip halt.

21.3.6 PRS-triggered Transmissions

If a transmission must be started on an event with very little delay, the PRS system can be used to trigger the transmission. The PRS channel to use as a trigger can be selected using PRSSEL in PRS_USARTn_TRIGGER. When a positive edge is detected on this signal, the receiver is enabled if RXTEN in USARTn_TRIGCTRL is set, and the transmitter is enabled if TXTEN in USARTn_TRIGCTRL is set. Only one signal input is supported by the USART.

The AUTOTX feature can also be enabled via PRS. If an external SPI device sets a pin high when there is data to be read from the device, this signal can be routed to the USART through the PRS system and be used to make the USART clock data out of the external device. If AUTOTXTEN in USARTn_TRIGCTRL is set, the USART will transmit data whenever the PRS signal selected by PRS_USARTn_TRIGGER is high given that there is enough room in the RX buffer for the chosen frame size. Note that if there is no data in the TX buffer when using AUTOTX, the TX underflow interrupt will be set.

AUTOTXTEN can also be combined with TXTEN to make the USART transmit a command to the external device prior to clocking out data. To do this, disable TX using the TXDIS command, load the TX buffer with the command and enable AUTOTXTEN and TXTEN. When the selected PRS input goes high, the USART will now transmit the loaded command, and then continue clocking out while both the PRS input is high and there is room in the RX buffer

21.3.7 PRS RX Input

The USART can be configured to receive clock directly from a PRS channel by setting RXPRSEN in USARTn_CTRLX. The PRS channel used is selected using PRSSEL in PRS_USARTn_RX.

21.3.8 PRS CLK Input

The USART can be configured to receive clock directly from a PRS channel by setting CLKPRSEN in USARTn_CTRLX. The PRS channel used is selected using PRSSEL in PRS_USARTn_CLK. This is useful in synchronous slave mode and can together with RX PRS input be used to input data from PRS.

21.3.9 DMA Support

The USART has full DMA support. The DMA controller can write to the transmit buffer using the registers USARTn_TXDATA, USARTn_TXDATAX, USARTn_TXDOUBLE and USARTn_TXDOUBLEX, and it can read from the receive buffer using the registers USARTn_RXDATA, USARTn_RXDATAX, USARTn_RXDOUBLE and USARTn_RXDOUBLEX. This enables single byte transfers, 9 bit data + control/status bits, double byte and double byte + control/status transfers both to and from the USART.

A request for the DMA controller to read from the USART receive buffer can come from the following source:

- Data available in the receive buffer
- Data available in the receive buffer and data is for the RIGHT I2S channel. Only used in I2S mode.

A write request can come from one of the following sources:

- Transmit buffer and shift register empty. No data to send.
- Transmit buffer has room for more data. This does not check the TXBIL for half full. For DMA use, it is either full or empty.
- Transmit buffer has room for RIGHT I2S data. Only used in I2S mode

Even though there are two sources for write requests to the DMA, only one should be used at a time, since the requests from both sources are cleared even though only one of the requests are used.

In some cases, it may be sensible to temporarily stop DMA access to the USART when an error such as a framing error has occurred. This is enabled by setting ERRSDMA in USARTn_CTRL.

Note: For Synchronous mode full duplex operation, if both receive buffer and transmit buffer are served by DMA, to make sure receive buffer is not overflowed the settings below should be followed.

- The DMA channel that serves receive buffer should have higher priority than the DMA channel that serves transmit buffer.
- TXBL should be used as write request for transmit buffer DMA channel.
- IGNORESREQ should be set for both DMA channel.

21.3.10 Timer

In addition to the TX sequence timer, there is a versatile 8 bit timer that can generate up to three event pulses. These pulses can be used to create timing for a variety of uses such as RX timeout, break detection, response timeout, and RX enable delay. Transmission delay, CS setup, inter-character spacing, and CS hold use the TX sequence counter. The TX sequencer counter can use the three 8 bit compare values or preset values for delays. There is one general counter with three comparators. Each comparator has a start source, a stop source, a restart enable, and a timer compare value. The start source enables the comparator, resets the counter, and starts the counter. If the counter is already running, the start source will reset the counter and restart it.

Any comparator could start the counter using the same start source but have different timing events programmed into TCMPVALn in USARTn_TIMECMPn. The TCMP0, TCMP1, or TCMP2 events can be preempted by using the comparator stop source to disable the comparator before the counter reaches TCMPVAL0, TCMPVAL1, or TCMPVAL2. If one comparator gets disabled while the other comparator is still enabled, the counter continues counting. By default the counter will count up to 256 and stop unless a RESTARTEN is set in one of the USARTn_TIMECMPn registers. By using RESTARTEN and an interval programmed into TCMPVAL, an interval timer can be set up. The TSTART field needs to be changed to DISABLE to stop the interval timer. The timer stops running once all of the comparators are disabled. If a comparator's start and stop sources both trigger the same cycle, the TCMPn event triggers, the comparator stays enabled, and the counter begins counting from zero.

The TXDELAY, CSSETUP, ICS, and CSHOLD in USARTn_TIMING are used to program start of transmission delay, chip select setup delay, inter-character space, and chip select hold delay. Either a preset value of 0, 1, 2, 3, or 7 can be used for any of these delays; or the value in TCMPVALn may be used to set the delay. Using the preset values leaves the TCMPVALn free for other uses. The same TCMPVALn may be used for multiple events that require the same timing. The transmit sequencer's counter can run in parallel with the timer's counter. The counters and controls are shown in Figure 21.25 USART Timer Block Diagram on page 584.

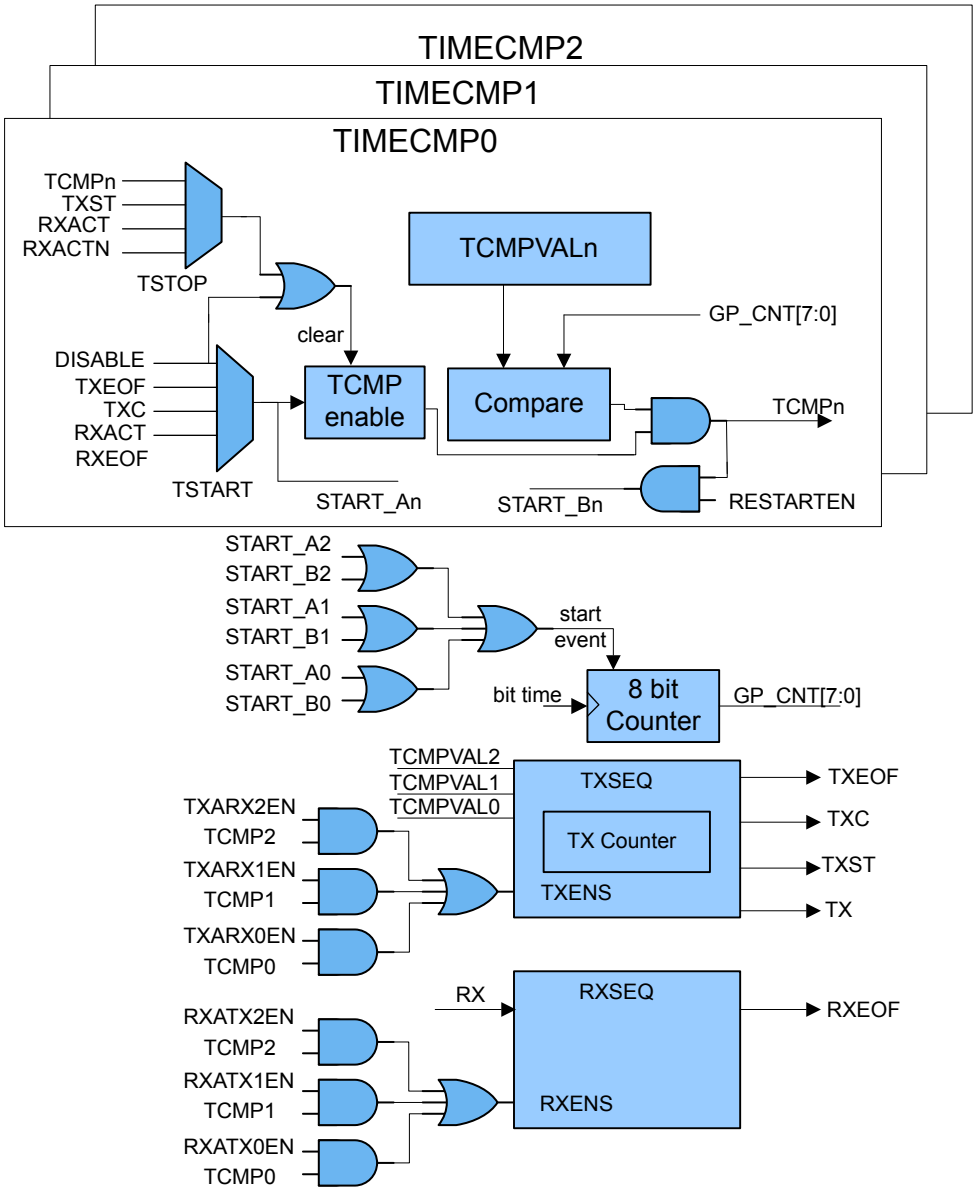


Figure 21.25. USART Timer Block Diagram

The following sections will go into more details on programming the various usage cases.

Table 21.10. USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn

Application	TSTARTn	TSTOPn	TCMPVALn	Other
Response Timeout	TSTART0 = TXEOF	TSTOP0 = RXACT	TCMPVAL0 = 0x08	TCMP0 in USARTn_IEN
Receiver Timeout	TSTART1 = RXEOF	TSTOP1 = RXACT	TCMPVAL1 = 0x08	TCMP1 in USARTn_IEN
Large Receiver Timeout	TSTART1 = RXEOF, TCMP1	TSTOP1 = RXACT	TCMPVAL1 = 0xFF	TCMP1 in USARTn_IEN; TIME-RRESTARTED in USARTn_STA-TUS; RESTART1EN in USARTn_TIMECMP1

Application	TSTARTn	TSTOPn	TCMPVALn	Other
Break Detect	TSTART1 = RXACT	TSTOP1 = RXACTN	TCMPVAL1 = 0x0C	TCMP1 in USARTn_IEN
TX delayed start of transmission and CS setup	TSTART0 = DISABLE, TSTART1 = DISABLE	TSTOP0 = TCMP0, TSTOP1 = TCMP1	TCMPVAL0 = 0x04, TCMPVAL1 = 0x02	TXDELAY = TCMP0, CSSETUP = TCMP1 in USARTn_TIMING; AUTOCS in USARTn_CTRL
TX inter-character spacing	TSTART2 = DISABLE	TSTOP2 = TCMP2	TCMPVAL2 = 0x03	ICS = TCMP2 in USARTn_TIMING; AUTOCS in USARTn_CTRL
TX Chip Select End Delay	TSTART1 = DISABLE	TSTOP1 = TCMP1	TCMPVAL1 = 0x04	CSHOLD = TCMP1 in USARTn_TIMING; AUTOCS in USARTn_CTRL
Response Delay	TSTART1 = RXEOF	TSTOP1 = TCMP1	TCMPVAL1 = 0x08	TXARX1EN in USARTn_TRIGCTRL
Combined TX and RX Example	TSTART1 = RXEOF, TSTART0 = TXEOF	TSTOP1 = TCMP1, TSTOP0 = TCMP0	TCMPVAL1 = 0x1C, TCMPVAL0 = 0x10	TXARX1EN, RXATX0EN in USARTn_TRIGCTRL; CSSETUP = 0x7, CSHOLD = 0x3 in USARTn_TIMING
Combined Delayed TX and Receiver Timeout Example	TSTART0 = TCMPVAL0, TSTART1 = RXEOF	TSTOP0 = RXACTN, TSTOP1 = RXACT	TCMPVAL0 = 0x20, TCMPVAL1 = 0x0C	TXARX0EN in USARTn_TRIGCTRL; TCMP0 in USARTn_IEN

Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 shows some examples of how the USART timer can be programmed for various applications. The following sections will describe more details for each applications shown in the table.

21.3.10.1 Response Timeout

Response Timeout is when a UART master sends a frame and expects the slave to respond within a certain number of baud-times. Refer to Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 for specific register settings. Comparator 0 will be looking for TX end of frame to use as the timer start source. For this example, a receiver start of frame RXACT has not been detected for 8 baud-times, and the TCMP0 interrupt in USARTn_IF is set. If an RX start bit is detected before the 8 baud-times, comparator 0 is disabled before the TCMP0 event can trigger.

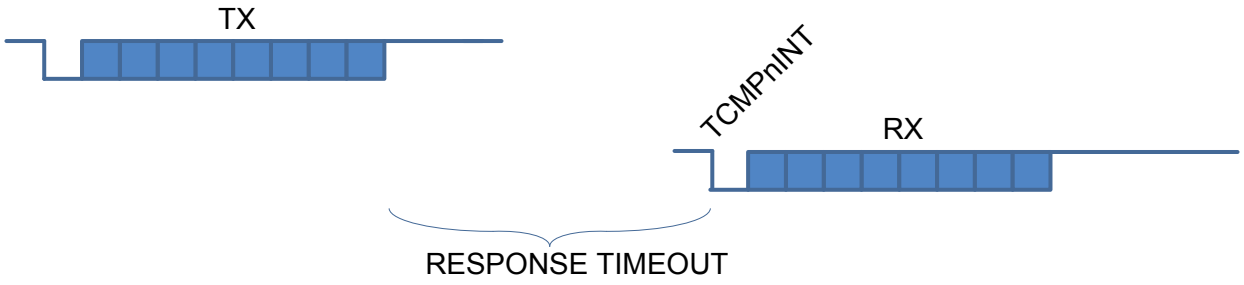


Figure 21.26. USART Response Timeout

21.3.10.2 RX Timeout

A receiver timeout function can be implemented by using the RX end of frame to start comparator 1 and look for the RX start bit RXACT to disable the comparator. See Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 for details on setting up this example. As long as the next RX start bit occurs before the counter reaches the comparator 1 value TCMPVAL1, the interrupt will not get set. In this example the RX Timeout was set to 8 baud-times. To get an RX timeout larger than 256 baud-times, RESTART1EN in USARTn_TIMER can used to restart the counter when it reaches TCMPVAL1. By setting TCMPVAL1 in USARTn_TIMING to 0xFF, an interrupt will be generated after 256 baud-times. An interrupt service routine can then increment a memory location until the desired timeout is reached. Once the RX start bit is detected, comparator 1 will be disabled. If TIMERRESTARTED in USARTn_STATUS is clear, the TCMP1 interrupt is the first interrupt after RXEOF.

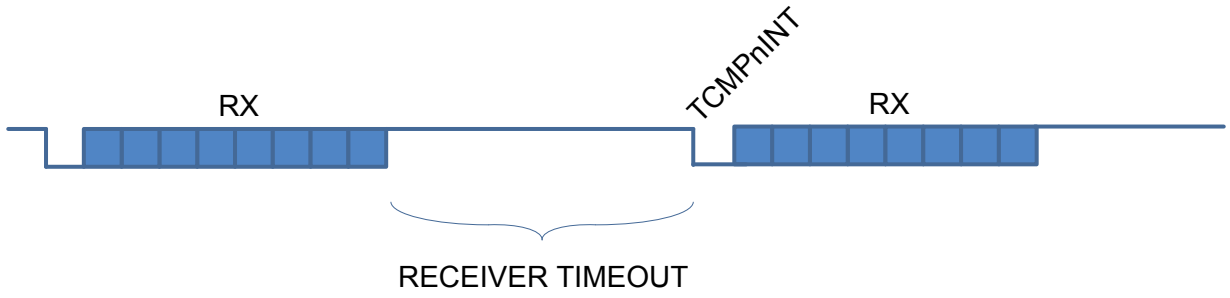


Figure 21.27. USART RX Timeout

21.3.10.3 Break Detect

LIN bus and half-duplex UARTs can take advantage of the timer configured for break detection where RX is held low for a number of baud-times to indicate a break condition. Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 shows the settings for this mode. Each time RX is active (default of low) such as for a start bit, the timer begins counting. If the counter reaches 12 baud-times before RX goes to inactive RXACTN (default of high), an interrupt is asserted.

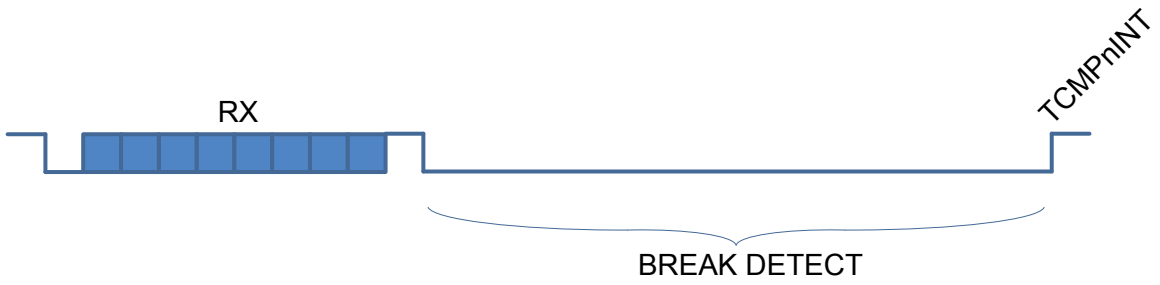


Figure 21.28. USART Break Detection

21.3.10.4 TX Start Delay

Some applications may require a delay before the start of transmission. This example in Figure 21.29 USART TXSEQ Timing on page 587 shows the TXSEQ timer used to delay the start of transmission by 4 baud times before the start of CS, and by 2 baud times with CS asserted. See Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 for details on how to configure this mode. The TX sequencer could be enabled on PRS and start the TXSEQ counter running for 4 baud times as programmed in TCMPVAL0. Then CS is asserted for 2 baud times before the transmitter begins sending TX data. TXDELAY in USARTn_TIMING is the initial delay before any CS assertion, and CSSETUP is the delay during CS assertion. There are several small preset timing values such as 1, 2, 3, or 7 that can be used for some of the TX sequencer timing which leaves TCMPVAL0, TCMPVAL1, and TCMPVAL2 free for other uses.

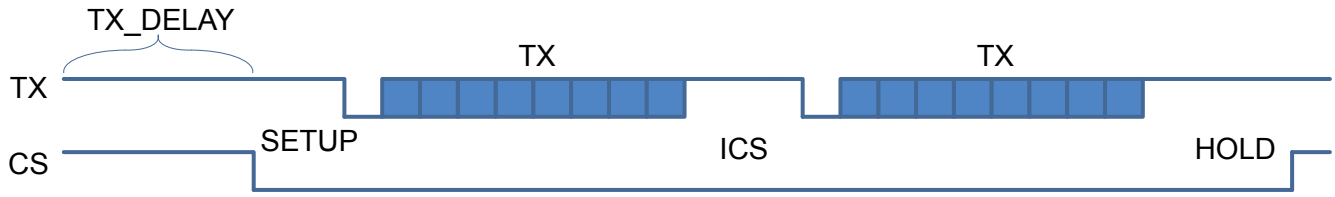


Figure 21.29. USART TXSEQ Timing

21.3.10.5 Inter-Character Space

In addition to delaying the start of frame transmission, it is sometimes necessary to also delay the time between each transmit character (inter-character space). After the first transmission, the inter-character space will delay the start of all subsequent transmissions until the transmit buffer is empty. See Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 for details on setting up this example. For this example in Figure 21.29 USART TXSEQ Timing on page 587 ICS is set to TCMP2 in USARTn_TIMING. To keep CS asserted during the inter-character space, set AUTOCS in USARTn_CTRL. There are a few small pre-set timing values provided for TX sequence timing. Using these preset timing values can free up the TCMPVALn for other uses. For this example, the inter-character space is set to 0x03 and a preset value could be used.

21.3.10.6 TX Chip Select End Delay

The assertion of CS can be extended after the final character of the frame by using CSHOLD in USARTn_TIMING. See Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 for details on setting up this example. AUTOCS in USARTn_CTRL needs to be set to extend the CS assertion after the last TX character is transmitted as shown in Figure 21.29 USART TXSEQ Timing on page 587.

21.3.10.7 Response Delay

A response delay can be used to hold off the transmitter until a certain number of baud-times after the RX frame. See Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 for details on setting up this example. TXARX1EN in USARTn_TRIGCTRL tells the TX sequencer to trigger after RX EOF plus tcmp1val baud times.

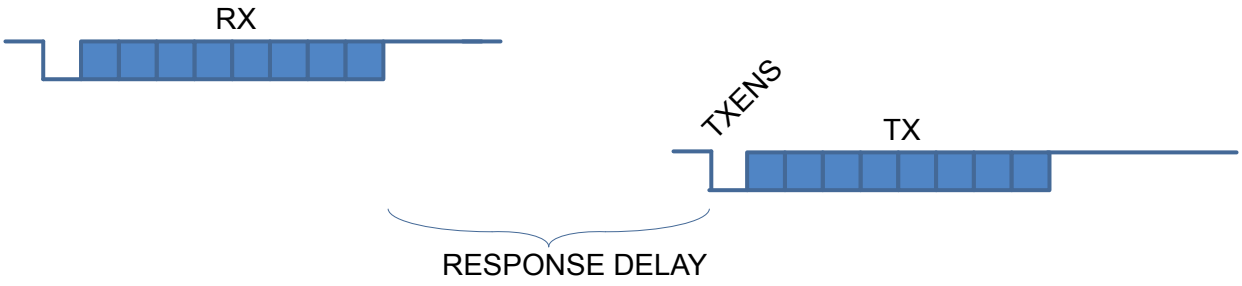


Figure 21.30. USART Response Delay

21.3.10.8 Combined TX and RX Example

This example describes how to alternate between TX and RX frames. This has a 28 baud-time space after RX and a 16 baud-time space after TX. The TSTART1 in USARTn_TIMECMP1 is set to RXEOF which uses the the receiver end of frame to start the timer. The TSTOP1 is set to TCMP1 to generate an event after 28 baud times. Set TXARX1EN in USARTn_TRIGCTRL, and the transmitter is held off until 28 baud times. TCMPVAL in USARTn_TIMECMP1 is set to 0x1C for 28 baud times. By setting TSTART0 in USARTn_TIMECMP0 to TXEOF, the timer will be started after the transmission has completed. RXATX0EN in USARTn_TRIGCTRL is used to delay enabling of the receiver until 16 baud times after the transmitter has completed. Write 0x10 into TCMPVAL of USARTn_TIMECMP0 for a 16 baud time delay. CS is also asserted 7 baud-times before start of transmission by setting CSSETUP to 0x7 in USARTn_TIMING. To keep CS asserted for 3 baud-times after transmission completes, CSHOLD is set to 0x3 in USARTn_TIMING. See Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 for details on setting up this example.

21.3.10.9 Combined TX delay and RX break detect

This example describes how to delay TX transmission after an RX frame and how to have a break condition signal an interrupt. See Table 21.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 584 for details on setting up this example. The TX delay is set up by using transmit after RX, TXARX0EN in USARTn_TRIGCTRL to start the timer. TSTART0 in USARTn_TIMECMP0 is set to RXEOF which enables the transitter of the timer delay. For this example TCMPVAL in USARTn_TIMECMP0 is set to 0x20 to create a 32 baud-time delay between the end of the RX frame and the start of the TX frame. The break detect is configured by setting TSTART1 to RXACT to detect the start bit, and setting TSTOP1 to RXACTN to detect RX going high. In this case the interrupt asserts after RX stays low for 12 baud-times, so TCMPVAL1 is set to 0x0C.

21.3.10.10 Other Stop Conditions

There is also a timer stop on TX start using the TXST setting in TSTOP of USARTn_TIMECMPn. This can be used to see that the DMA has not written to the TXBUFFER for a given time.

21.3.11 Interrupts

The interrupts generated by the USART are combined into two interrupt vectors. Interrupts related to reception are assigned to one interrupt vector, and interrupts related to transmission are assigned to the other. Separating the interrupts in this way allows different priorities to be set for transmission and reception interrupts.

The transmission interrupt vector groups the transmission-related interrupts generated by the following interrupt flags:

- TXC
- TXBL
- TXOF
- CCF
- TXIDLE

The reception interrupt on the other hand groups the reception-related interrupts, triggered by the following interrupt flags:

- RXDATAV
- RXFULL
- RXOF
- RXUF
- PERR
- FERR
- MPAF
- SSM
- TCMPn

If USART interrupts are enabled, an interrupt will be made if one or more of the interrupt flags in USART_IF and their corresponding bits in USART_IEN are set.

21.3.12 IrDA Modulator/ Demodulator

The IrDA modulator implements the physical layer of the IrDA specification, which is necessary for communication over IrDA. The modulator takes the signal output from the USART module, and modulates it before it leaves the USART. In the same way, the input signal is demodulated before it enters the actual USART module. The modulator implements the original Rev. 1.0 physical layer and one high speed extension which supports speeds from 2.4 kbps to 1.152 Mbps.

The data from and to the USART is represented in a NRZ (Non Return to Zero) format, where the signal value is at the same level through the entire bit period. For IrDA, the required format is RZI (Return to Zero Inverted), a format where a “1” is signalled by holding the line low, and a “0” is signalled by a short high pulse. An example is given in Figure 21.31 USART Example RZI Signal for a given Asynchronous USART Frame on page 589.

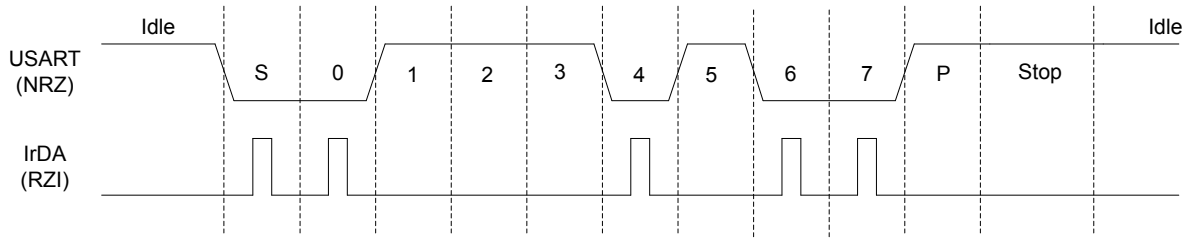


Figure 21.31. USART Example RZI Signal for a given Asynchronous USART Frame

The IrDA module is enabled by setting IREN. The USART transmitter output and receiver input is then routed through the IrDA modulator.

The width of the pulses generated by the IrDA modulator is set by configuring IRPW in USARTn_IRCTRL. Four pulse widths are available, each defined relative to the configured bit period as listed in Table 21.11 USART IrDA Pulse Widths on page 589.

Table 21.11. USART IrDA Pulse Widths

IRPW	Pulse width OVS=0	Pulse width OVS=1	Pulse width OVS=2	Pulse width OVS=3
00	1/16	1/8	1/6	1/4
01	2/16	2/8	2/6	N/A
10	3/16	3/8	N/A	N/A
11	4/16	N/A	N/A	N/A

By default, no filter is enabled in the IrDA demodulator. A filter can be enabled by setting IRFILT in USARTn_IRCTRL. When the filter is enabled, an incoming pulse has to last for 4 consecutive clock cycles to be detected by the IrDA demodulator.

Note that by default, the idle value of the USART data signal is high. This means that the IrDA modulator generates negative pulses, and the IrDA demodulator expects negative pulses. To make the IrDA module use RZI signalling, both TXINV and RXINV in USARTn_CTRL must be set.

21.4 USART Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	USART_IPVERSION	R	IPVERSION
0x004	USART_EN	RW	USART Enable
0x008	USART_CTRL	RW	Control Register
0x00C	USART_FRAME	RW	USART Frame Format Register
0x010	USART_TRIGCTRL	RW	USART Trigger Control register
0x014	USART_CMD	W	Command Register
0x018	USART_STATUS	RH	USART Status Register
0x01C	USART_CLKDIV	RWH	Clock Control Register
0x020	USART_RXDATAX	RH	RX Buffer Data Extended Register
0x024	USART_RXDATA	RH	RX Buffer Data Register
0x028	USART_RXDOUBLEX	RH	RX Buffer Double Data Extended Register
0x02C	USART_RXDOUBLE	RH	RX FIFO Double Data Register
0x030	USART_RXDATAXP	RH	RX Buffer Data Extended Peek Register
0x034	USART_RXDOUBLEXP	RH	RX Buffer Double Data Extended Peek R...
0x038	USART_TXDATAX	W	TX Buffer Data Extended Register
0x03C	USART_TXDATA	W	TX Buffer Data Register
0x040	USART_TXDOUBLEX	W	TX Buffer Double Data Extended Register
0x044	USART_TXDOUBLE	W	TX Buffer Double Data Register
0x048	USART_IF	RWH INTFLAG	Interrupt Flag Register
0x04C	USART_IEN	RW	Interrupt Enable Register
0x050	USART_IRCTRL	RW	IrDA Control Register
0x054	USART_I2SCTRL	RW	I2S Control Register
0x058	USART_TIMING	RW	Timing Register
0x05C	USART_CTRLX	RW	Control Register Extended
0x060	USART_TIMECMP0	RW	Used to generate interrupts and vario...
0x064	USART_TIMECMP1	RW	Used to generate interrupts and vario...
0x068	USART_TIMECMP2	RW	Used to generate interrupts and vario...
0x1000	USART_IPVERSION_SET	R	IPVERSION
0x1004	USART_EN_SET	RW	USART Enable
0x1008	USART_CTRL_SET	RW	Control Register
0x100C	USART_FRAME_SET	RW	USART Frame Format Register
0x1010	USART_TRIGCTRL_SET	RW	USART Trigger Control register
0x1014	USART_CMD_SET	W	Command Register
0x1018	USART_STATUS_SET	RH	USART Status Register
0x101C	USART_CLKDIV_SET	RWH	Clock Control Register

Offset	Name	Type	Description
0x1020	USART_RXDATAX_SET	RH	RX Buffer Data Extended Register
0x1024	USART_RXDATA_SET	RH	RX Buffer Data Register
0x1028	USART_RXDOUBLEX_SET	RH	RX Buffer Double Data Extended Register
0x102C	USART_RXDOUBLE_SET	RH	RX FIFO Double Data Register
0x1030	USART_RXDATAXP_SET	RH	RX Buffer Data Extended Peek Register
0x1034	USART_RXDOUBLEXP_SET	RH	RX Buffer Double Data Extended Peek R...
0x1038	USART_TXDATAX_SET	W	TX Buffer Data Extended Register
0x103C	USART_TXDATA_SET	W	TX Buffer Data Register
0x1040	USART_TXDOUBLEX_SET	W	TX Buffer Double Data Extended Register
0x1044	USART_TXDOUBLE_SET	W	TX Buffer Double Data Register
0x1048	USART_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x104C	USART_IEN_SET	RW	Interrupt Enable Register
0x1050	USART_IRCTRL_SET	RW	IrDA Control Register
0x1054	USART_I2SCTRL_SET	RW	I2S Control Register
0x1058	USART_TIMING_SET	RW	Timing Register
0x105C	USART_CTRLX_SET	RW	Control Register Extended
0x1060	USART_TIMECMP0_SET	RW	Used to generate interrupts and vario...
0x1064	USART_TIMECMP1_SET	RW	Used to generate interrupts and vario...
0x1068	USART_TIMECMP2_SET	RW	Used to generate interrupts and vario...
0x2000	USART_IPVERSION_CLR	R	IPVERSION
0x2004	USART_EN_CLR	RW	USART Enable
0x2008	USART_CTRL_CLR	RW	Control Register
0x200C	USART_FRAME_CLR	RW	USART Frame Format Register
0x2010	USART_TRIGCTRL_CLR	RW	USART Trigger Control register
0x2014	USART_CMD_CLR	W	Command Register
0x2018	USART_STATUS_CLR	RH	USART Status Register
0x201C	USART_CLKDIV_CLR	RWH	Clock Control Register
0x2020	USART_RXDATAX_CLR	RH	RX Buffer Data Extended Register
0x2024	USART_RXDATA_CLR	RH	RX Buffer Data Register
0x2028	USART_RXDOUBLEX_CLR	RH	RX Buffer Double Data Extended Register
0x202C	USART_RXDOUBLE_CLR	RH	RX FIFO Double Data Register
0x2030	USART_RXDATAXP_CLR	RH	RX Buffer Data Extended Peek Register
0x2034	USART_RXDOUBLEXP_CLR	RH	RX Buffer Double Data Extended Peek R...
0x2038	USART_TXDATAX_CLR	W	TX Buffer Data Extended Register
0x203C	USART_TXDATA_CLR	W	TX Buffer Data Register
0x2040	USART_TXDOUBLEX_CLR	W	TX Buffer Double Data Extended Register
0x2044	USART_TXDOUBLE_CLR	W	TX Buffer Double Data Register

Offset	Name	Type	Description
0x2048	USART_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x204C	USART_IEN_CLR	RW	Interrupt Enable Register
0x2050	USART_IRCTRL_CLR	RW	IrDA Control Register
0x2054	USART_I2SCTRL_CLR	RW	I2S Control Register
0x2058	USART_TIMING_CLR	RW	Timing Register
0x205C	USART_CTRLX_CLR	RW	Control Register Extended
0x2060	USART_TIMECMP0_CLR	RW	Used to generate interrupts and vario...
0x2064	USART_TIMECMP1_CLR	RW	Used to generate interrupts and vario...
0x2068	USART_TIMECMP2_CLR	RW	Used to generate interrupts and vario...
0x3000	USART_IPVERSION_TGL	R	IPVERSION
0x3004	USART_EN_TGL	RW	USART Enable
0x3008	USART_CTRL_TGL	RW	Control Register
0x300C	USART_FRAME_TGL	RW	USART Frame Format Register
0x3010	USART_TRIGCTRL_TGL	RW	USART Trigger Control register
0x3014	USART_CMD_TGL	W	Command Register
0x3018	USART_STATUS_TGL	RH	USART Status Register
0x301C	USART_CLKDIV_TGL	RWH	Clock Control Register
0x3020	USART_RXDATAX_TGL	RH	RX Buffer Data Extended Register
0x3024	USART_RXDATA_TGL	RH	RX Buffer Data Register
0x3028	USART_RXDOUBLEX_TGL	RH	RX Buffer Double Data Extended Register
0x302C	USART_RXDOUBLE_TGL	RH	RX FIFO Double Data Register
0x3030	USART_RXDATAXP_TGL	RH	RX Buffer Data Extended Peek Register
0x3034	USART_RXDOUBLEXP_TGL	RH	RX Buffer Double Data Extended Peek R...
0x3038	USART_TXDATAX_TGL	W	TX Buffer Data Extended Register
0x303C	USART_TXDATA_TGL	W	TX Buffer Data Register
0x3040	USART_TXDOUBLEX_TGL	W	TX Buffer Double Data Extended Register
0x3044	USART_TXDOUBLE_TGL	W	TX Buffer Double Data Register
0x3048	USART_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x304C	USART_IEN_TGL	RW	Interrupt Enable Register
0x3050	USART_IRCTRL_TGL	RW	IrDA Control Register
0x3054	USART_I2SCTRL_TGL	RW	I2S Control Register
0x3058	USART_TIMING_TGL	RW	Timing Register
0x305C	USART_CTRLX_TGL	RW	Control Register Extended
0x3060	USART_TIMECMP0_TGL	RW	Used to generate interrupts and vario...
0x3064	USART_TIMECMP1_TGL	RW	Used to generate interrupts and vario...
0x3068	USART_TIMECMP2_TGL	RW	Used to generate interrupts and vario...

21.5 USART Register Description

21.5.1 USART_IPVERSION - IPVERSION

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IPVERSION**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

21.5.2 USART_EN - USART Enable

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**USART Enable**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

21.5.3 USART_CTRL - Control Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0	0x0			0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0		0x0		0x0	0x0	0x0	0x0	0x0
Access	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW		RW	RW	RW	RW	RW
Name	SMSDELAY	MVDIS	AUTOTX	BYTESWAP			SSSEARLY	ERRSTX	ERRSRX	ERRSDMA	BIT8DV	SKIPPERRF	SCRETRANS	SCMODE	AUTOTRI	AUTOCS	CSINV	TXINV	RXINV	TXBIL	CSMA	MSBF	CLKPHA	CLKPOL		OVS		MPAB	MPM	CCEN	LOOPBK	SYNC

Bit	Name	Reset	Access	Description
31	SMSDELAY	0x0	RW	**Synchronous Master Sample Delay**
	Delay Synchronous Master sample point to the next setup edge to improve timing and allow communication at higher speeds			
30	MVDIS	0x0	RW	**Majority Vote Disable**
	Disable majority vote for 16x, 8x and 6x oversampling modes.			
29	AUTOTX	0x0	RW	**Always Transmit When RX Not Full**
	Transmits as long as RX is not full. If TX is empty, underflows are generated.			
28	BYTESWAP	0x0	RW	**Byteswap In Double Accesses**
	Set to switch the order of the bytes in double accesses.			

Value	Mode	Description
0	DISABLE	Normal byte order
1	ENABLE	Byte order swapped

27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	SSSEARLY	0x0	RW	**Synchronous Slave Setup Early**
	Setup data on sample edge in synchronous slave mode to improve MOSI setup time			
24	ERRSTX	0x0	RW	**Disable TX On Error**
	When set, the transmitter is disabled on framing and parity errors (asynchronous mode only) in the receiver.			

Value	Mode	Description
0	DISABLE	Received framing and parity errors have no effect on transmitter
1	ENABLE	Received framing and parity errors disable the transmitter

23	ERRSRX	0x0	RW	**Disable RX On Error**
	When set, the receiver is disabled on framing and parity errors (asynchronous mode only).			

Value	Mode	Description
0	DISABLE	Framing and parity errors have no effect on receiver
1	ENABLE	Framing and parity errors disable the receiver

Bit	Name	Reset	Access	Description
22	ERRSDMA	0x0	RW	**Halt DMA On Error**

When set, DMA requests will be cleared on framing and parity errors (asynchronous mode only).

Value	Mode	Description
0	DISABLE	Framing and parity errors have no effect on DMA requests from the USART
1	ENABLE	DMA requests from the USART are blocked while the PERR or FERR interrupt flags are set

Bit	Name	Reset	Access	Description
21	BIT8DV	0x0	RW	**Bit 8 Default Value**

The default value of the 9th bit. If 9-bit frames are used, and an 8-bit write operation is done, leaving the 9th bit unspecified, the 9th bit is set to the value of BIT8DV.

Bit	Name	Reset	Access	Description
20	SKIPPERRF	0x0	RW	**Skip Parity Error Frames**

When set, the receiver discards frames with parity errors (asynchronous mode only). The PERR interrupt flag is still set.

Bit	Name	Reset	Access	Description
19	SCRETRANS	0x0	RW	**SmartCard Retransmit**

When in SmartCard mode, a NACK'ed frame will be kept in the shift register and retransmitted if the transmitter is still enabled.

Bit	Name	Reset	Access	Description
18	SCMODE	0x0	RW	**SmartCard Mode**

Use this bit to enable or disable SmartCard mode.

Bit	Name	Reset	Access	Description
17	AUTOTRI	0x0	RW	**Automatic TX Tristate**

When enabled, TXTRI is set by hardware whenever the transmitter is idle, and TXTRI is cleared by hardware when transmission starts.

Value	Mode	Description
0	DISABLE	The output on U(S)n_TX when the transmitter is idle is defined by TXINV
1	ENABLE	U(S)n_TX is tristated whenever the transmitter is idle

Bit	Name	Reset	Access	Description
16	AUTOCS	0x0	RW	**Automatic Chip Select**

When enabled, the output on USn_CS will be activated one baud-period before transmission starts, and deactivated when transmission ends.

Bit	Name	Reset	Access	Description
15	CSINV	0x0	RW	**Chip Select Invert**

Default value is active low. This affects both the selection of external slaves, as well as the selection of the microcontroller as a slave.

Value	Mode	Description
0	DISABLE	Chip select is active low
1	ENABLE	Chip select is active high

Bit	Name	Reset	Access	Description
14	TXINV	0x0	RW	**Transmitter output Invert**

The output from the USART transmitter can optionally be inverted by setting this bit.

Value	Mode	Description
0	DISABLE	Output from the transmitter is passed unchanged to U(S)n_TX
1	ENABLE	Output from the transmitter is inverted before it is passed to U(S)n_TX

Bit	Name	Reset	Access	Description
13	RXINV	0x0	RW	**Receiver Input Invert**

Setting this bit will invert the input to the USART receiver.

Value	Mode	Description
0	DISABLE	Input is passed directly to the receiver
1	ENABLE	Input is inverted before it is passed to the receiver

Bit	Name	Reset	Access	Description
12	TXBIL	0x0	RW	**TX Buffer Interrupt Level**

Determines the interrupt and status level of the transmit buffer.

Value	Mode	Description
0	EMPTY	TXBL and the TXBL interrupt flag are set when the transmit buffer becomes empty. TXBL is cleared when the buffer becomes nonempty.
1	HALFFULL	TXBL and TXBLIF are set when the transmit buffer goes from full to half-full or empty. TXBL is cleared when the buffer becomes full.

Bit	Name	Reset	Access	Description
11	CSMA	0x0	RW	**Action On Slave-Select In Master Mode**

This register determines the action to be performed when slave-select is configured as an input and driven low while in master mode.

Value	Mode	Description
0	NOACTION	No action taken
1	GOTOSLAVEMODE	Go to slave mode

Bit	Name	Reset	Access	Description
10	MSBF	0x0	RW	**Most Significant Bit First**

Decides whether data is sent with the least significant bit first, or the most significant bit first.

Value	Mode	Description
0	DISABLE	Data is sent with the least significant bit first
1	ENABLE	Data is sent with the most significant bit first

Bit	Name	Reset	Access	Description
9	CLKPHA	0x0	RW	**Clock Edge For Setup/Sample**

Determines where data is set-up and sampled according to the bus clock when in synchronous mode.

Value	Mode	Description
0	SAMPLELEADING	Data is sampled on the leading edge and set-up on the trailing edge of the bus clock in synchronous mode
1	SAMPLETRAILING	Data is set-up on the leading edge and sampled on the trailing edge of the bus clock in synchronous mode

Bit	Name	Reset	Access	Description
8	CLKPOL	0x0	RW	**Clock Polarity**

Determines the clock polarity of the bus clock used in synchronous mode.

Value	Mode	Description
0	IDLELOW	The bus clock used in synchronous mode has a low base value

Bit	Name	Reset	Access	Description
	1	IDLEHIGH		The bus clock used in synchronous mode has a high base value
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:5	OVS	0x0	RW	**Oversampling**
	Sets the number of clock periods in a UART bit-period. More clock cycles gives better robustness, while less clock cycles gives better performance.			
	Value	Mode		Description
	0	X16		Regular UART mode with 16X oversampling in asynchronous mode
	1	X8		Double speed with 8X oversampling in asynchronous mode
	2	X6		6X oversampling in asynchronous mode
	3	X4		Quadruple speed with 4X oversampling in asynchronous mode
4	MPAB	0x0	RW	**Multi-Processor Address-Bit**
	Defines the value of the multi-processor address bit. An incoming frame with its 9th bit equal to the value of this bit marks the frame as a multi-processor address frame.			
3	MPM	0x0	RW	**Multi-Processor Mode**
	Multi-processor mode uses the 9th bit of the USART frames to tell whether the frame is an address frame or a data frame.			
	Value	Mode		Description
	0	DISABLE		The 9th bit of incoming frames has no special function
	1	ENABLE		An incoming frame with the 9th bit equal to MPAB will be loaded into the receive buffer regardless of RXBLOCK and will result in the MPAB interrupt flag being set
2	CCEN	0x0	RW	**Collision Check Enable**
	Enables collision checking on data when operating in half duplex modus.			
	Value	Mode		Description
	0	DISABLE		Collision check is disabled
	1	ENABLE		Collision check is enabled. The receiver must be enabled for the check to be performed
1	LOOPBK	0x0	RW	**Loopback Enable**
	Allows the receiver to be connected directly to the USART transmitter for loopback and half duplex communication.			
	Value	Mode		Description
	0	DISABLE		The receiver is connected to and receives data from U(S)n_RX
	1	ENABLE		The receiver is connected to and receives data from U(S)n_TX
0	SYNC	0x0	RW	**USART Synchronous Mode**
	Determines whether the USART is operating in asynchronous or synchronous mode.			
	Value	Mode		Description

Bit	Name	Reset	Access	Description
	0	DISABLE		The USART operates in asynchronous mode
	1	ENABLE		The USART operates in synchronous mode

21.5.4 USART_FRAME - USART Frame Format Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x1				0x0						0x5			
Access																			RW				RW						RW			
Name																			STOPBITS				PARITY						DATABITS			

Bit	Name	Reset	Access	Description
31:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	STOPBITS	0x1	RW	**Stop-Bit Mode**

Determines the number of stop-bits used.

Value	Mode	Description
0	HALF	The transmitter generates a half stop bit. Stop-bits are not verified by receiver
1	ONE	One stop bit is generated and verified
2	ONEANDAHALF	The transmitter generates one and a half stop bit. The receiver verifies the first stop bit
3	TWO	The transmitter generates two stop bits. The receiver checks the first stop-bit only

11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	PARITY	0x0	RW	**Parity-Bit Mode**

Determines whether parity bits are enabled, and whether even or odd parity should be used. Only available in asynchronous mode.

Value	Mode	Description
0	NONE	Parity bits are not used
2	EVEN	Even parity are used. Parity bits are automatically generated and checked by hardware.
3	ODD	Odd parity is used. Parity bits are automatically generated and checked by hardware.

7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	DATABITS	0x5	RW	**Data-Bit Mode**

This register sets the number of data bits in a USART frame.

Value	Mode	Description
1	FOUR	Each frame contains 4 data bits
2	FIVE	Each frame contains 5 data bits

Bit	Name	Reset	Access	Description
	3	SIX		Each frame contains 6 data bits
	4	SEVEN		Each frame contains 7 data bits
	5	EIGHT		Each frame contains 8 data bits
	6	NINE		Each frame contains 9 data bits
	7	TEN		Each frame contains 10 data bits
	8	ELEVEN		Each frame contains 11 data bits
	9	TWELVE		Each frame contains 12 data bits
	10	THIRTEEN		Each frame contains 13 data bits
	11	FOURTEEN		Each frame contains 14 data bits
	12	FIFTEEN		Each frame contains 15 data bits
	13	SIXTEEN		Each frame contains 16 data bits

21.5.5 USART_TRIGCTRL - USART Trigger Control register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																				0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0				
Access																				RW	RW	RW	RW	RW	RW	RW	RW	RW				
Name																				RXATX2EN	RXATX1EN	RXATX0EN	TXARX2EN	TXARX1EN	TXARX0EN	AUTOTXTEN	TXTEN	RXTEN				

Bit	Name	Reset	Access	Description
31:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	RXATX2EN	0x0	RW	**Enable Receive Trigger after TX end of f**
	When set, a TX end of frame will trigger the receiver after a TCMPVAL2 baud-time delay			
11	RXATX1EN	0x0	RW	**Enable Receive Trigger after TX end of f**
	When set, a TX end of frame will trigger the receiver after a TCMPVAL1 baud-time delay			
10	RXATX0EN	0x0	RW	**Enable Receive Trigger after TX end of f**
	When set, a TX end of frame will trigger the receiver after a TCMPVAL0 baud-time delay			
9	TXARX2EN	0x0	RW	**Enable Transmit Trigger after RX End of**
	When set, an RX end of frame will trigger the transmitter after TCMP2VAL bit times to force a minimum response delay			
8	TXARX1EN	0x0	RW	**Enable Transmit Trigger after RX End of**
	When set, an RX end of frame will trigger the transmitter after TCMP1VAL bit times to force a minimum response delay			
7	TXARX0EN	0x0	RW	**Enable Transmit Trigger after RX End of**
	When set, an RX end of frame will trigger the transmitter after TCMP0VAL bit times to force a minimum response delay			
6	AUTOTXTEN	0x0	RW	**AUTOTX Trigger Enable**
	When set, AUTOTX is enabled as long as the PRS channel selected by TSEL has a high value			
5	TXTEN	0x0	RW	**Transmit Trigger Enable**
	When set, the PRS channel selected by TSEL sets TXEN, enabling the transmitter on positive trigger edges.			
4	RXTEN	0x0	RW	**Receive Trigger Enable**
	When set, the PRS channel selected by TSEL sets RXEN, enabling the receiver on positive trigger edges.			
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

21.5.6 USART_CMD - Command Register

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																					W	W	W	W	W	W	W	W	W	W	W	W
Name																					CLEARRX	CLEARTX	TXTRIDIS	TXTRIEN	RXBLOCKDIS	RXBLOCKEN	MASTERDIS	MASTEREN	TXDIS	TXEN	RXDIS	RXEN

Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	CLEARRX	0x0	W	**Clear RX**
	Set to clear receive buffer and the RX shift register.			
10	CLEARTX	0x0	W	**Clear TX**
	Set to clear transmit buffer and the TX shift register.			
9	TXTRIDIS	0x0	W	**Transmitter Tristate Disable**
	Disables tristating of the transmitter output.			
8	TXTRIEN	0x0	W	**Transmitter Tristate Enable**
	Tristates the transmitter output.			
7	RXBLOCKDIS	0x0	W	**Receiver Block Disable**
	Set to clear RXBLOCK, resulting in all incoming frames being loaded into the receive buffer.			
6	RXBLOCKEN	0x0	W	**Receiver Block Enable**
	Set to set RXBLOCK, resulting in all incoming frames being discarded.			
5	MASTERDIS	0x0	W	**Master Disable**
	Set to disable master mode, clearing the MASTER status bit and putting the USART in slave mode.			
4	MASTEREN	0x0	W	**Master Enable**
	Set to enable master mode, setting the MASTER status bit. Master mode should not be enabled while TXENS is set to 1. To enable both master and TX mode, write MASTEREN before TXEN, or enable them both in the same write operation.			
3	TXDIS	0x0	W	**Transmitter Disable**
	Set to disable transmission.			
2	TXEN	0x0	W	**Transmitter Enable**
	Set to enable data transmission.			
1	RXDIS	0x0	W	**Receiver Disable**
	Set to disable data reception. If a frame is under reception when the receiver is disabled, the incoming frame is discarded.			
0	RXEN	0x0	W	**Receiver Enable**
	Set to activate data reception on U(S)n_RX.			

21.5.7 USART_STATUS - USART Status Register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0			0x0	0x1	0x0	0x0	0x0	0x0	0x0	0x0	0x1	0x0	0x0	0x0	0x0	0x0	0x0
Access															R			R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name															TXBUFCNT			TIMERRESTARTED	TXIDLE	RXFULLRIGHT	RXDATAVRIGHT	TXBSRIGHT	TXBDRIGHT	RXFULL	RXDATAV	TXBL	TXC	TXTRI	RXBLOCK	MASTER	TXENS	RXENS

Bit	Name	Reset	Access	Description
31:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	TXBUFCNT	0x0	R	**TX Buffer Count**
	Count of TX buffer entry 0, entry 1, and TX shift register. For large frames, the count is only of TX buffer entry 0 and the TX shifter register.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	TIMERRESTARTED	0x0	R	**The USART Timer restarted itself**
	When the timer is restarting itself on each TCMP event, a TIMERRESTARTED value of 0x0 indicates the first TCMP event in the sequence of multiple TCMP events. Any non TCMP timer start events will clear TIMERRESTARTED. When there is a TCMP interrupt and TIMERRESTARTED is 0x0, an interrupt service routine can set a TCMP event counter variable in memory to 0x1 to indicate the first TCMP interrupt of the sequence.			
13	TXIDLE	0x1	R	**TX Idle**
	Set when TX idle			
12	RXFULLRIGHT	0x0	R	**RX Full of Right Data**
	When set, the entire RX buffer contains right data. Only used in I2S mode			
11	RXDATAVRIGHT	0x0	R	**RX Data Right**
	When set, reading RXDATA or RXDATAX gives right data. Else left data is read. Only used in I2S mode			
10	TXBSRIGHT	0x0	R	**TX Buffer Expects Single Right Data**
	When set, the TX buffer expects at least a single right data. Else it expects left data. Only used in I2S mode			
9	TXBDRIGHT	0x0	R	**TX Buffer Expects Double Right Data**
	When set, the TX buffer expects double right data. Else it may expect a single right data or left data. Only used in I2S mode			
8	RXFULL	0x0	R	**RX FIFO Full**
	Set when the RXFIFO is full. Cleared when the receive buffer is no longer full. When this bit is set, there is still room for one more frame in the receive shift register.			
7	RXDATAV	0x0	R	**RX Data Valid**
	Set when data is available in the receive buffer. Cleared when the receive buffer is empty.			
6	TXBL	0x1	R	**TX Buffer Level**

Bit	Name	Reset	Access	Description
	Indicates the level of the transmit buffer. If TXBIL is 0x0, TXBL is set whenever the transmit buffer is completely empty. Otherwise TXBL is set whenever the TX Buffer becomes half full.			
5	TXC	0x0	R	**TX Complete**
	Set when a transmission has completed and no more data is available in the transmit buffer and shift register. Cleared when data is written to the transmit buffer.			
4	TXTRI	0x0	R	**Transmitter Tristated**
	Set when the transmitter is tristated, and cleared when transmitter output is enabled. If AUTOTRI in USARTn_CTRL is set this bit is always read as 0.			
3	RXBLOCK	0x0	R	**Block Incoming Data**
	When set, the receiver discards incoming frames. An incoming frame will not be loaded into the receive buffer if this bit is set at the instant the frame has been completely received.			
2	MASTER	0x0	R	**SPI Master Mode**
	Set when the USART operates as a master. Set using the MASTEREN command and clear using the MASTERDIS command.			
1	TXENS	0x0	R	**Transmitter Enable Status**
	Set when the transmitter is enabled.			
0	RXENS	0x0	R	**Receiver Enable Status**
	Set when the receiver is enabled.			

21.5.8 USART_CLKDIV - Clock Control Register

Offset	Bit Position																														
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																		0x0													
Access	RW																		RW													
Name	AUTOBAUDEN																		DIV													

Bit	Name	Reset	Access	Description
31	AUTOBAUDEN	0x0	RW	**AUTOBAUD detection enable**
	Detects the baud rate based on receiving a 0x55 frame (0x00 for IrDA). This is used in Asynchronous mode.			
30:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:3	DIV	0x0	RW	**Fractional Clock Divider**
	Specifies the fractional clock divider for the USART. Setting AUTOBAUDEN in USARTn_CLKDIV will overwrite the DIV field.			
2:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

21.5.9 USART_RXDATAX - RX Buffer Data Extended Register

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0	0x0						0x0								
Access																	R	R						R								
Name																	FERR	PERR						RXDATA								

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	FERR	0x0	R	**Data Framing Error**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERR	0x0	R	**Data Parity Error**
	Set if data in buffer has a parity error (asynchronous mode only).			
13:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	RXDATA	0x0	R	**RX Data**
	Use this register to access data read from the USART. Buffer is cleared on read access.			

21.5.10 USART_RXDATA - RX Buffer Data Register

Offset	Bit Position																															
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									RXDATA							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	RXDATA	0x0	R	**RX Data**
	Use this register to access data read from USART. Buffer is cleared on read access. Only the 8 LSB can be read using this register.			

21.5.11 USART_RXDOUBLEX - RX Buffer Double Data Extended Register

Offset	Bit Position																														
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0						0x0									0x0	0x0						0x0								
Access	R	R						R									R	R						R								
Name	FERR1	PERR1						RXDATA1									FERR0	PERR0						RXDATA0								

Bit	Name	Reset	Access	Description
31	FERR1	0x0	R	**Data Framing Error 1**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
30	PERR1	0x0	R	**Data Parity Error 1**
	Set if data in buffer has a parity error (asynchronous mode only).			
29:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24:16	RXDATA1	0x0	R	**RX Data 1**
	Second frame read from buffer.			
15	FERR0	0x0	R	**Data Framing Error 0**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERR0	0x0	R	**Data Parity Error 0**
	Set if data in buffer has a parity error (asynchronous mode only).			
13:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	RXDATA0	0x0	R	**RX Data 0**
	First frame read from buffer.			

21.5.12 USART_RXDOUBLE - RX FIFO Double Data Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0								0x0							
Access																	R								R							
Name																	RXDATA1								RXDATA0							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	RXDATA1	0x0	R	**RX Data 1**
	Second frame read from buffer.			
7:0	RXDATA0	0x0	R	**RX Data 0**
	First frame read from buffer.			

21.5.13 USART_RXDATAXP - RX Buffer Data Extended Peek Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0	0x0							0x0							
Access																	R	R							R							
Name																	FERRP	PERRP							RXDATAP							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	FERRP	0x0	R	**Data Framing Error Peek**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERRP	0x0	R	**Data Parity Error Peek**
	Set if data in buffer has a parity error (asynchronous mode only).			
13:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	RXDATAP	0x0	R	**RX Data Peek**
	Use this register to access data read from the USART.			

21.5.14 USART_RXDOUBLEXP - RX Buffer Double Data Extended Peek R...

Offset	Bit Position																															
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0						0x0									0x0	0x0						0x0								
Access	R	R						R									R	R						R								
Name	FERRP1	PERRP1						RXDATAP1									FERRP0	PERRP0						RXDATAP0								

Bit	Name	Reset	Access	Description
31	FERRP1	0x0	R	**Data Framing Error 1 Peek**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
30	PERRP1	0x0	R	**Data Parity Error 1 Peek**
	Set if data in buffer has a parity error (asynchronous mode only).			
29:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24:16	RXDATAP1	0x0	R	**RX Data 1 Peek**
	Second frame read from FIFO.			
15	FERRP0	0x0	R	**Data Framing Error 0 Peek**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERRP0	0x0	R	**Data Parity Error 0 Peek**
	Set if data in buffer has a parity error (asynchronous mode only).			
13:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	RXDATAP0	0x0	R	**RX Data 0 Peek**
	First frame read from FIFO.			

21.5.15 USART_TXDATAX - TX Buffer Data Extended Register

Offset	Bit Position																															
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0	0x0	0x0	0x0	0x0			0x0								
Access																	W	W	W	W	W			W								
Name																	RXENAT	TXDISAT	TXBREAK	TXTRIAT	UBRXAT			TXDATAX								

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	RXENAT	0x0	W	**Enable RX After Transmission**
	Set to enable reception after transmission.			
14	TXDISAT	0x0	W	**Clear TXEN After Transmission**
	Set to disable transmitter and release data bus directly after transmission.			
13	TXBREAK	0x0	W	**Transmit Data As Break**
	Set to send data as a break. Recipient will see a framing error or a break condition depending on its configuration and the value of TXDATA.			
12	TXTRIAT	0x0	W	**Set TXTRI After Transmission**
	Set to tristate transmitter by setting TXTRI after transmission.			
11	UBRXAT	0x0	W	**Unblock RX After Transmission**
	Set to clear RXBLOCK after transmission, unblocking the receiver.			
10:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	TXDATAX	0x0	W	**TX Data**
	Use this register to write data to the USART. If TXEN is set, a transfer will be initiated at the first opportunity.			

21.5.16 USART_TXDATA - TX Buffer Data Register

Offset	Bit Position																													
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									TXDATA							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	TXDATA	0x0	W	**TX Data** This frame will be added to TX buffer. Only 8 LSB can be written using this register. 9th bit and control bits will be cleared.

21.5.17 USART_TXDOUBLEX - TX Buffer Double Data Extended Register

Offset	Bit Position																														
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0	0x0	0x0			0x0									0x0	0x0	0x0	0x0	0x0			0x0								
Access	W	W	W	W	W			W									W	W	W	W	W			W								
Name	RXENAT1	TXDISAT1	TXBREAK1	TXTRIAT1	UBRXAT1			TXDATA1									RXENAT0	TXDISAT0	TXBREAK0	TXTRIAT0	UBRXAT0			TXDATA0								

Bit	Name	Reset	Access	Description
31	RXENAT1	0x0	W	**Enable RX After Transmission**
	Set to enable reception after transmission.			
30	TXDISAT1	0x0	W	**Clear TXEN After Transmission**
	Set to disable transmitter and release data bus directly after transmission.			
29	TXBREAK1	0x0	W	**Transmit Data As Break**
	Set to send data as a break. Recipient will see a framing error or a break condition depending on its configuration and the value of USARTn_TXDATA.			
28	TXTRIAT1	0x0	W	**Set TXTRI After Transmission**
	Set to tristate transmitter by setting TXTRI after transmission.			
27	UBRXAT1	0x0	W	**Unblock RX After Transmission**
	Set clear RXBLOCK after transmission, unblocking the receiver.			
26:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24:16	TXDATA1	0x0	W	**TX Data**
	Second frame to write to FIFO.			
15	RXENAT0	0x0	W	**Enable RX After Transmission**
	Set to enable reception after transmission.			
14	TXDISAT0	0x0	W	**Clear TXEN After Transmission**
	Set to disable transmitter and release data bus directly after transmission.			
13	TXBREAK0	0x0	W	**Transmit Data As Break**
	Set to send data as a break. Recipient will see a framing error or a break condition depending on its configuration and the value of TXDATA.			
12	TXTRIAT0	0x0	W	**Set TXTRI After Transmission**
	Set to tristate transmitter by setting TXTRI after transmission.			
11	UBRXAT0	0x0	W	**Unblock RX After Transmission**
	Set clear RXBLOCK after transmission, unblocking the receiver.			
10:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	TXDATA0	0x0	W	**TX Data**
	First frame to write to buffer.			

21.5.18 USART_TXDOUBLE - TX Buffer Double Data Register

Offset	Bit Position																													
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0								0x0							
Access																	W								W							
Name																	TXDATA1								TXDATA0							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	TXDATA1	0x0	W	**TX Data**
	Second frame to write to buffer.			
7:0	TXDATA0	0x0	W	**TX Data**
	First frame to write to buffer.			

21.5.19 USART_IF - Interrupt Flag Register

Offset	Bit Position																															
0x048	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x1	0x0
Access																RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																TCMP2	TCMP1	TCMP0	TXIDLE	CCF	SSM	MPAF	FERR	PERR	TXUF	TXOF	RXUF	RXOF	RXFULL	RXDATAV	TXBL	TXC

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	TCMP2	0x0	RW	**Timer comparator 2 Interrupt Flag**
	Set when the timer reaches the comparator 2 value, TCMP2.			
15	TCMP1	0x0	RW	**Timer comparator 1 Interrupt Flag**
	Set when the timer reaches the comparator 1 value, TCMP1.			
14	TCMP0	0x0	RW	**Timer comparator 0 Interrupt Flag**
	Set when the Timer reaches the comparator 0 value, TCMP0.			
13	TXIDLE	0x0	RW	**TX Idle Interrupt Flag**
	Set when TX goes idle. At this point, transmission has ended			
12	CCF	0x0	RW	**Collision Check Fail Interrupt Flag**
	Set when a collision check notices an error in the transmitted data.			
11	SSM	0x0	RW	**Slave-Select In Master Mode Interrupt Fl**
	Set when the device is selected as a slave when in master mode.			
10	MPAF	0x0	RW	**Multi-Processor Address Frame Interrupt**
	Set when a multi-processor address frame is detected.			
9	FERR	0x0	RW	**Framing Error Interrupt Flag**
	Set when a frame with a framing error is received while RXBLOCK is cleared.			
8	PERR	0x0	RW	**Parity Error Interrupt Flag**
	Set when a frame with a parity error (asynchronous mode only) is received while RXBLOCK is cleared.			
7	TXUF	0x0	RW	**TX Underflow Interrupt Flag**
	Set when operating as a synchronous slave, no data is available in the transmit buffer when the master starts transmission of a new frame.			
6	TXOF	0x0	RW	**TX Overflow Interrupt Flag**
	Set when a write is done to the transmit buffer while it is full. The data already in the transmit buffer is preserved.			
5	RXUF	0x0	RW	**RX Underflow Interrupt Flag**
	Set when trying to read from the receive buffer when it is empty.			
4	RXOF	0x0	RW	**RX Overflow Interrupt Flag**
	Set when data is incoming while the receive shift register is full. The data previously in the shift register is lost.			

Bit	Name	Reset	Access	Description
3	RXFULL	0x0	RW	**RX Buffer Full Interrupt Flag**
	Set when the receive buffer becomes full.			
2	RXDATAV	0x0	RW	**RX Data Valid Interrupt Flag**
	Set when data becomes available in the receive buffer.			
1	TXBL	0x1	RW	**TX Buffer Level Interrupt Flag**
	Set when buffer becomes empty if buffer level is set to 0x0, or when the number of empty TX buffer elements equals specified buffer level.			
0	TXC	0x0	RW	**TX Complete Interrupt Flag**
	This interrupt is set after a transmission when both the TX buffer and shift register are empty.			

21.5.20 USART_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x04C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																TCMP2	TCMP1	TCMP0	TXIDLE	CCF	SSM	MPAF	FERR	PERR	TXUF	TXOF	RXUF	RXOF	RXFULL	RXDATAV	TXBL	TXC

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	TCMP2	0x0	RW	**Timer comparator 2 Interrupt Enable**
	Set when the timer reaches the comparator 2 value, TCMP2.			
15	TCMP1	0x0	RW	**Timer comparator 1 Interrupt Enable**
	Set when the timer reaches the comparator 1 value, TCMP1.			
14	TCMP0	0x0	RW	**Timer comparator 0 Interrupt Enable**
	Set when the Timer reaches the comparator 0 value, TCMP0.			
13	TXIDLE	0x0	RW	**TX Idle Interrupt Enable**
	Set when TX goes idle. At this point, transmission has ended			
12	CCF	0x0	RW	**Collision Check Fail Interrupt Enable**
	Set when a collision check notices an error in the transmitted data.			
11	SSM	0x0	RW	**Slave-Select In Master Mode Interrupt Fl**
	Set when the device is selected as a slave when in master mode.			
10	MPAF	0x0	RW	**Multi-Processor Address Frame Interrupt**
	Set when a multi-processor address frame is detected.			
9	FERR	0x0	RW	**Framing Error Interrupt Enable**
	Set when a frame with a framing error is received while RXBLOCK is cleared.			
8	PERR	0x0	RW	**Parity Error Interrupt Enable**
	Set when a frame with a parity error (asynchronous mode only) is received while RXBLOCK is cleared.			
7	TXUF	0x0	RW	**TX Underflow Interrupt Enable**
	Set when operating as a synchronous slave, no data is available in the transmit buffer when the master starts transmission of a new frame.			
6	TXOF	0x0	RW	**TX Overflow Interrupt Enable**
	Set when a write is done to the transmit buffer while it is full. The data already in the transmit buffer is preserved.			
5	RXUF	0x0	RW	**RX Underflow Interrupt Enable**
	Set when trying to read from the receive buffer when it is empty.			
4	RXOF	0x0	RW	**RX Overflow Interrupt Enable**
	Set when data is incoming while the receive shift register is full. The data previously in the shift register is lost.			

Bit	Name	Reset	Access	Description
3	RXFULL	0x0	RW	**RX Buffer Full Interrupt Enable**
	Set when the receive buffer becomes full.			
2	RXDATAV	0x0	RW	**RX Data Valid Interrupt Enable**
	Set when data becomes available in the receive buffer.			
1	TXBL	0x0	RW	**TX Buffer Level Interrupt Enable**
	Set when buffer becomes empty if buffer level is set to 0x0, or when the number of empty TX buffer elements equals specified buffer level.			
0	TXC	0x0	RW	**TX Complete Interrupt Enable**
	This interrupt is set after a transmission when both the TX buffer and shift register are empty.			

21.5.21 USART_IRCTRL - IrDA Control Register

Offset	Bit Position																															
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0		0x0
Access																													RW	RW		RW
Name																													IRFILT	IRPW		IREN

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	IRFILT	0x0	RW	**IrDA RX Filter**
	Set to enable filter on IrDA demodulator.			

Value	Mode	Description
0	DISABLE	No filter enabled
1	ENABLE	Filter enabled. IrDA pulse must be high for at least 5 consecutive clock cycles to be detected

2:1	IRPW	0x0	RW	**IrDA TX Pulse Width**
	Configure the pulse width generated by the IrDA modulator as a fraction of the configured USART bit period.			

Value	Mode	Description
0	ONE	IrDA pulse width is 1/16 for OVS=0 and 1/8 for OVS=1
1	TWO	IrDA pulse width is 2/16 for OVS=0 and 2/8 for OVS=1
2	THREE	IrDA pulse width is 3/16 for OVS=0 and 3/8 for OVS=1
3	FOUR	IrDA pulse width is 4/16 for OVS=0 and 4/8 for OVS=1

0	IREN	0x0	RW	**Enable IrDA Module**
	Enable IrDA module and rout USART signals through it.			

21.5.22 USART_I2SCTRL - I2S Control Register

Offset	Bit Position																													
0x054	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0					0x0	0x0	0x0	0x0	0x0
Access																							RW					RW	RW	RW	RW	RW
Name																							FORMAT					DELAY	DMASPLIT	JUSTIFY	MONO	EN

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	FORMAT	0x0	RW	**I2S Word Format**
	Configure the data-width used internally for I2S data			

Value	Mode	Description
0	W32D32	32-bit word, 32-bit data
1	W32D24M	32-bit word, 32-bit data with 8 lsb masked
2	W32D24	32-bit word, 24-bit data
3	W32D16	32-bit word, 16-bit data
4	W32D8	32-bit word, 8-bit data
5	W16D16	16-bit word, 16-bit data
6	W16D8	16-bit word, 8-bit data
7	W8D8	8-bit word, 8-bit data

7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	DELAY	0x0	RW	**Delay on I2S data**
	Set to add a one-cycle delay between a transition on the word-clock and the start of the I2S word. Should be set for standard I2S format			
3	DMASPLIT	0x0	RW	**Separate DMA Request For Left/Right Data**
	When set DMA requests for right-channel data are put on the TXBLRIGHT and RXDATAVRIGHT DMA requests.			
2	JUSTIFY	0x0	RW	**Justification of I2S Data**
	Determines whether the I2S data is left or right justified			

Value	Mode	Description
0	LEFT	Data is left-justified
1	RIGHT	Data is right-justified

1	MONO	0x0	RW	**Stero or Mono**
	Switch between stereo and mono mode. Set for mono			
0	EN	0x0	RW	**Enable I2S Mode**

Bit	Name	Reset	Access	Description
	Set the U(S)ART in I2S mode.			

21.5.23 USART_TIMING - Timing Register

Offset	Bit Position																															
0x058	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0				0x0				0x0				0x0																		
Access		RW				RW				RW				RW																		
Name		CSHOLD				ICS				CSSETUP				TXDELAY																		

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:28	CSHOLD	0x0	RW	**Chip Select Hold**

Chip Select will be asserted after the end of frame transmission. When using TCMPn, normally set TIMECMPn_TSTART to DISABLE to stop general timer and to prevent unwanted interrupts.

Value	Mode	Description
0	ZERO	Disable CS being asserted after the end of transmission
1	ONE	CS is asserted for 1 baud-times after the end of transmission
2	TWO	CS is asserted for 2 baud-times after the end of transmission
3	THREE	CS is asserted for 3 baud-times after the end of transmission
4	SEVEN	CS is asserted for 7 baud-times after the end of transmission
5	TCMP0	CS is asserted after the end of transmission for TCMPVAL0 baud-times
6	TCMP1	CS is asserted after the end of transmission for TCMPVAL1 baud-times
7	TCMP2	CS is asserted after the end of transmission for TCMPVAL2 baud-times

27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:24	ICS	0x0	RW	**Inter-character spacing**

Inter-character spacing after each TX frame while the TX buffer is not empty. When using USART_TIMECMPn, normally set TSTART to DISABLE to stop general timer and to prevent unwanted interrupts.

Value	Mode	Description
0	ZERO	There is no space between charcters
1	ONE	Create a space of 1 baud-times before start of transmission
2	TWO	Create a space of 2 baud-times before start of transmission
3	THREE	Create a space of 3 baud-times before start of transmission
4	SEVEN	Create a space of 7 baud-times before start of transmission
5	TCMP0	Create a space of before the start of transmission for TCMPVAL0 baud-times

Bit	Name	Reset	Access	Description
	6	TCMP1		Create a space of before the start of transmission for TCMPVAL1 baud-times
	7	TCMP2		Create a space of before the start of transmission for TCMPVAL2 baud-times
23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	CSSETUP	0x0	RW	**Chip Select Setup**
	Chip Select will be asserted before the start of frame transmission. When using USART_TIMECMPn, normally set TSTART to DISABLE to stop general timer and to prevent unwanted interrupts.			
	Value	Mode		Description
	0	ZERO		CS is not asserted before start of transmission
	1	ONE		CS is asserted for 1 baud-times before start of transmission
	2	TWO		CS is asserted for 2 baud-times before start of transmission
	3	THREE		CS is asserted for 3 baud-times before start of transmission
	4	SEVEN		CS is asserted for 7 baud-times before start of transmission
	5	TCMP0		CS is asserted before the start of transmission for TCMPVAL0 baud-times
	6	TCMP1		CS is asserted before the start of transmission for TCMPVAL1 baud-times
	7	TCMP2		CS is asserted before the start of transmission for TCMPVAL2 baud-times
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	TXDELAY	0x0	RW	**TX frame start delay**
	Number of baud-times to delay the start of frame transmission. When using USART_TIMECMPn, normally set TSTART to DISABLE to stop general timer and to prevent unwanted interrupts.			
	Value	Mode		Description
	0	DISABLE		Disable - TXDELAY in USARTn_CTRL can be used for legacy
	1	ONE		Start of transmission is delayed for 1 baud-times
	2	TWO		Start of transmission is delayed for 2 baud-times
	3	THREE		Start of transmission is delayed for 3 baud-times
	4	SEVEN		Start of transmission is delayed for 7 baud-times
	5	TCMP0		Start of transmission is delayed for TCMPVAL0 baud-times
	6	TCMP1		Start of transmission is delayed for TCMPVAL1 baud-times
	7	TCMP2		Start of transmission is delayed for TCMPVAL2 baud-times
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

21.5.24 USART_CTRLX - Control Register Extended

Offset	Bit Position																													
0x05C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0								0x0				0x0	0x0	0x0	0x0
Access																	RW								RW				RW	RW	RW	RW
Name																	CLKPRSEN								RXPRSEN				RTSINV	CTSEN	CTSINV	DBGHALT

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	CLKPRSEN	0x0	RW	**PRS CLK Enable**
	When set, the PRS channel selected as input to CLK.			
14:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	RXPRSEN	0x0	RW	**PRS RX Enable**
	When set, the PRS channel selected as input to RX.			
6:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	RTSINV	0x0	RW	**RTS Pin Inversion**
	When set, the RTS pin polarity is inverted.			

Value	Mode	Description
0	DISABLE	The USn_RTS pin is low true
1	ENABLE	The USn_RTS pin is high true

Bit	Name	Reset	Access	Description
2	CTSEN	0x0	RW	**CTS Function enabled**
	When set, frames in the TXBUFn will not be sent until link partner asserts CTS. Any data in the TX shift register will continue transmitting, the next TXBUFn data will not load into the TX shift register			

Value	Mode	Description
0	DISABLE	Ingore CTS
1	ENABLE	Stop transmitting when CTS is negated

Bit	Name	Reset	Access	Description
1	CTSINV	0x0	RW	**CTS Pin Inversion**
	When set, the CTS pin polarity is inverted.			

Value	Mode	Description
0	DISABLE	The USn_CTS pin is low true
1	ENABLE	The USn_CTS pin is high true

Bit	Name	Reset	Access	Description
0	DBGHALT	0x0	RW	**Debug halt**

Bit	Name	Reset	Access	Description
	.			

Value	Mode	Description
0	DISABLE	Continue to transmit until TX buffer is empty
1	ENABLE	Negate RTS to stop link partner's transmission during debug HALT. NOTE** The core clock should be equal to or faster than the peripheral clock; otherwise, each single step could transmit multiple frames instead of just transmitting one frame.

21.5.25 USART_TIMECMP0 - Used to generate interrupts and vario...

Offset	Bit Position																															
0x060	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x0		0x0				0x0											0x0							
Access								RW		RW				RW											RW							
Name								RESTARTEN		TSTOP				TSTART											TCMPVAL							

Bit	Name	Reset	Access	Description
31:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	RESTARTEN	0x0	RW	**Restart Timer on TCMP0**
	Each TCMP0 event will reset and restart the timer			

Value	Mode	Description
0	DISABLE	Disable the timer restarting on TCMP0
1	ENABLE	Enable the timer restarting on TCMP0

23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	TSTOP	0x0	RW	**Source used to disable comparator 0**
	Select the source which disables comparator 0			

Value	Mode	Description
0	TCMP0	Comparator 0 is disabled when the counter equals TCMPVAL and triggers a TCMP0 event
1	TXST	Comparator 0 is disabled at TX start TX Engine
2	RXACT	Comparator 0 is disabled on RX going going Active (default: low)
3	RXACTN	Comparator 0 is disabled on RX going Inactive

19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	TSTART	0x0	RW	**Timer start source**
	Source used to start comparator 0 and timer			

Value	Mode	Description
0	DISABLE	Comparator 0 is disabled
1	TXEOF	Comparator 0 and timer are started at TX end of frame
2	TXC	Comparator 0 and timer are started at TX Complete
3	RXACT	Comparator 0 and timer are started at RX going going Active (default: low)

Bit	Name	Reset	Access	Description
	4	RXEOF		Comparator 0 and timer are started at RX end of frame
15:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	TCMPVAL	0x0	RW	**Timer comparator 0.** When the timer equals TCMPVAL, this signals a TCMP0 event and sets the TCMP0 flag. This event can also be used to enable various USART functionality. A value of 0x00 represents 256 baud times.

21.5.26 USART_TIMECMP1 - Used to generate interrupts and vario...

Offset	Bit Position																															
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x0		0x0				0x0											0x0							
Access								RW		RW				RW											RW							
Name								RESTARTEN		TSTOP				TSTART											TCMPVAL							

Bit	Name	Reset	Access	Description
31:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	RESTARTEN	0x0	RW	**Restart Timer on TCMP1**
	Each TCMP1 event will reset and restart the timer			

Value	Mode	Description
0	DISABLE	Disable the timer restarting on TCMP1
1	ENABLE	Enable the timer restarting on TCMP1

23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	TSTOP	0x0	RW	**Source used to disable comparator 1**
	Select the source which disables comparator 1			

Value	Mode	Description
0	TCMP1	Comparator 1 is disabled when the counter equals TCMPVAL and triggers a TCMP1 event
1	TXST	Comparator 1 is disabled at TX start TX Engine
2	RXACT	Comparator 1 is disabled on RX going going Active (default: low)
3	RXACTN	Comparator 1 is disabled on RX going Inactive

19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	TSTART	0x0	RW	**Timer start source**
	Source used to start comparator 1 and timer			

Value	Mode	Description
0	DISABLE	Comparator 1 is disabled
1	TXEOF	Comparator 1 and timer are started at TX end of frame
2	TXC	Comparator 1 and timer are started at TX Complete
3	RXACT	Comparator 1 and timer are started at RX going going Active (default: low)

Bit	Name	Reset	Access	Description
	4	RXEOF		Comparator 1 and timer are started at RX end of frame
15:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	TCMPVAL	0x0	RW	**Timer comparator 1.** When the timer equals TCMPVAL, this signals a TCMP1 event and sets the TCMP1 flag. This event can also be used to enable various USART functionality. A value of 0x00 represents 256 baud times.

21.5.27 USART_TIMECMP2 - Used to generate interrupts and vario...

Offset	Bit Position																															
0x068	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x0		0x0				0x0											0x0							
Access								RW		RW				RW											RW							
Name								RESTARTEN		TSTOP				TSTART											TCMPVAL							

Bit	Name	Reset	Access	Description
31:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	RESTARTEN	0x0	RW	**Restart Timer on TCMP2**
	Each TCMP2 event will reset and restart the timer			

Value	Mode	Description
0	DISABLE	Disable the timer restarting on TCMP2
1	ENABLE	Enable the timer restarting on TCMP2

23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	TSTOP	0x0	RW	**Source used to disable comparator 2**
	Select the source which disables comparator 2			

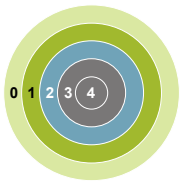
Value	Mode	Description
0	TCMP2	Comparator 2 is disabled when the counter equals TCMPVAL and triggers a TCMP2 event
1	TXST	Comparator 2 is disabled at TX start TX Engine
2	RXACT	Comparator 2 is disabled on RX going going Active (default: low)
3	RXACTN	Comparator 2 is disabled on RX going Inactive

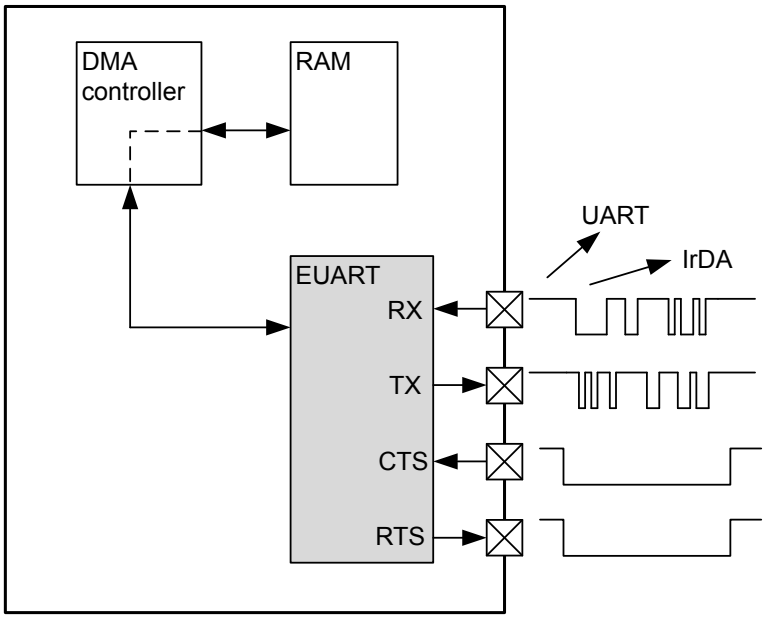
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	TSTART	0x0	RW	**Timer start source**
	Source used to start comparator 2 and timer			

Value	Mode	Description
0	DISABLE	Comparator 2 is disabled
1	TXEOF	Comparator 2 and timer are started at TX end of frame
2	TXC	Comparator 2 and timer are started at TX Complete
3	RXACT	Comparator 2 and timer are started at RX going going Active (default: low)

Bit	Name	Reset	Access	Description
	4	RXEOF		Comparator 2 and timer are started at RX end of frame
15:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	TCMPVAL	0x0	RW	**Timer comparator 2.** When the timer equals TCMPVAL, this signals a TCMP2 event and sets the TCMP2 flag. This event can also be used to enable various USART functionality. A value of 0x00 represents 256 baud times.

22. EUART - Enhanced Universal Asynchronous Receiver/Transmitter





Quick Facts

What?

The EUART handles high-speed UART and IrDA communication.

Why?

Serial communication is frequently used in embedded systems and the EUART allows efficient communication with a wide range of external devices.

How?

The EUART has a wide selection of operating modes, frame formats and baud rates. The multiprocessor mode allows the EUART to remain idle when not addressed. Triple buffering and DMA support makes high data-rates possible with minimal CPU intervention and it is possible to transmit and receive large frames while the MCU remains in EM1 Sleep. Lower-frequency operation in EM2 is supported on select I/O ports.

22.1 Introduction

The Enhanced Universal Asynchronous serial Receiver and Transmitter (EUART) is a very flexible serial I/O module. It supports full duplex asynchronous UART communication. It can also interface with IrDA devices.

22.2 Features

- Asynchronous (UART) communication
- Full duplex and half duplex
- Separate TX/RX enable
- Separate receive / transmit four-deep FIFOs, with additional separate shift registers
- Programmable baud rate, generated as a fractional division from the peripheral clock
- Max bit-rate
  - HF EM0/1 operation: peripheral clock rate/16, 8, 6, or 4
  - LF EM2 operation: 9600 baud from 32.768 kHz oscillator source
- Majority vote baud-reception
- False start-bit detection
- Break generation/detection
- Multi-processor mode
- Data can be transmitted LSB first or MSB first
- Configurable number of data bits, 4-9 (plus the parity bit, if enabled)
  - HW parity bit generation and check
- Configurable number of stop bits:
  - HF EM0/1 operation: 0.5, 1, 1.5, 2
  - LF EM2 operation: 1, 2
- HW collision detection
- IrDA support
  - HF EM0/1 operation: IrDA modulator
  - LF EM2 operation: Pulse extender, RX-only
- Separate interrupt vectors for receive and transmit interrupts
- Loopback mode
  - Half duplex communication
  - Communication debugging
- PRS RX input
- Hardware Flow Control
- Automatic Baud Rate Detection operating as HF EM0/1

22.3 Functional Description

An overview of the EUART module is shown in Figure 22.1 EUART Overview on page 631.

This section describes all posible EUART features. Please refer to the Device Datasheet to see what features a specific EUART instance supports.

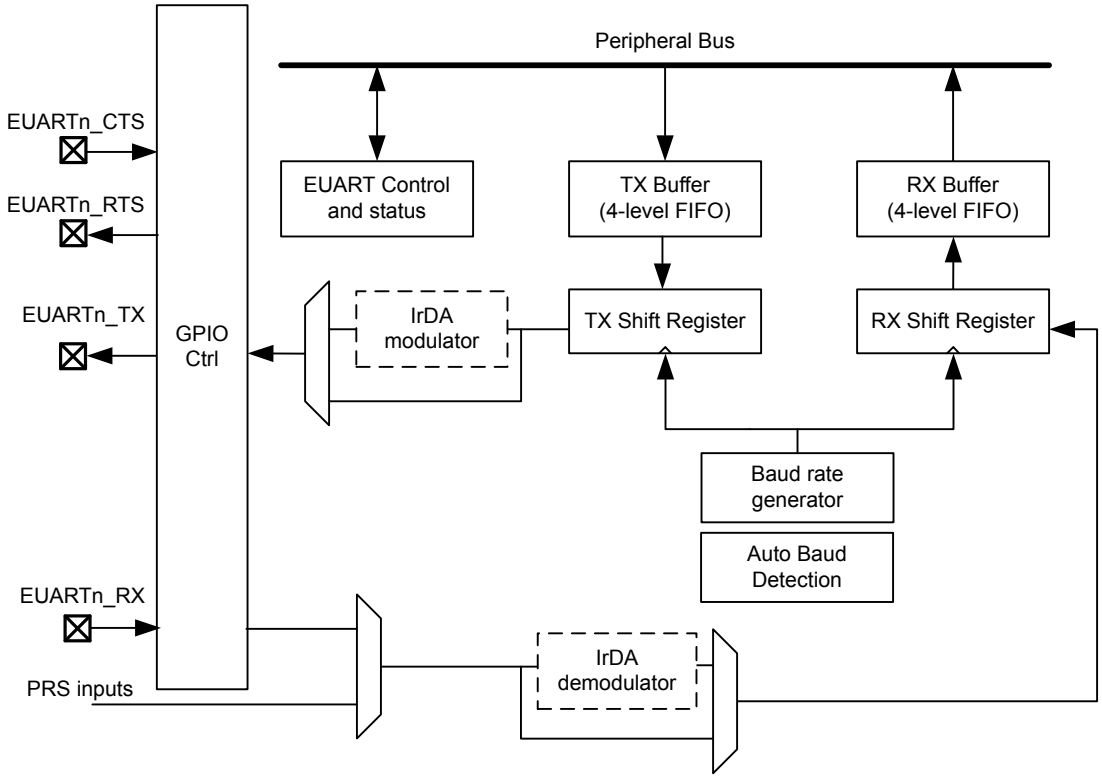


Figure 22.1. EUART Overview

22.3.1 Modes of Operation

The EUART may operate as either a high-speed peripheral running from a high-frequency clock source (HF mode, available in EM0 an EM1), or as a low-energy peripheral operating from a low-frequency clock source (LF mode, available in EM0, EM1, or EM2).

The EUART operates in HF mode when the EUARTCLK clock selected in CMU_UARTCLKCTRL_CLKSEL is EM01GRPACLK. The EUART operates in LF mode when the selected clock is EM23GRPACLK.

Baud rate generation differs between these two modes, and there are certain operational restrictions in LF mode discussed in this chapter. It is not generally useful to switch between modes on-the-fly in a single application.

22.3.2 Frame Format

The frame format used by the EUART consists of a set of data bits in addition to bits for synchronization and optionally a parity bit for error checking. A frame starts with one start-bit (S), where the line is driven low for one bit-period. This signals the start of a frame, and is used for synchronization. Following the start bit are 4 to 9 data bits and an optional parity bit. Finally, a number of stop-bits, where the line is driven high, end the frame. An example frame is shown in Figure 22.2 EUART Asynchronous Frame Format on page 632.

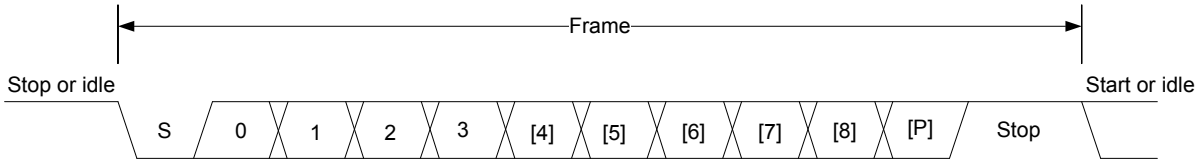


Figure 22.2. EUART Asynchronous Frame Format

The number of data bits in a frame is set by DATABITS in EUARTn_FRAMECFG, see Table 22.1 EUART Data Bits on page 632, and the number of stop-bits is set by STOPBITS in EUARTn_FRAMECFG, see Table 22.2 EUART Stop Bits on page 632. Whether or not a parity bit should be included, and whether it should be even or odd is defined by PARITY, also in EUARTn_FRAMECFG. For reliable communication, all parties of a transfer must agree on the frame format being used prior to the start of the transfer.

Table 22.1. EUART Data Bits

EUARTn_FRAMECFG_DATABITS [3:0]	Number of Data bits
0001	4
0010	5
0011	6
0100	7
0101	8 (Default)
0110	9

Table 22.2. EUART Stop Bits

EUARTn_FRAMECFG_STOPBITS [1:0]	Number of Stop bits
00	0.5 (HFCLK ONLY)
01	1 (Default)
10	1.5 (HFCLK ONLY)
11	2

The order in which the data bits are transmitted and received is defined by MSBF in EUARTn_CFG0. When MSBF is cleared, data in a frame is sent and received with the least significant bit first. When it is set, the most significant bit comes first.

The frame format used by the transmitter can be inverted by setting TXINV in EUARTn_CFG0, and the format expected by the receiver can be inverted by setting RXINV in EUARTn_CFG0. These bits affect the entire frame, not only the data bits. An inverted frame has a low idle state, a high start-bit, inverted data and parity bits, and low stop-bits.

22.3.3 Parity bit Calculation and Handling

When parity bits are enabled, hardware automatically calculates and inserts any parity bits into outgoing frames, and verifies the received parity bits in incoming frames. This is true for both asynchronous and synchronous modes, even though it is mostly used in asynchronous communication. The possible parity modes are defined in Table 22.3 EUART Parity Bits on page 633. When even parity is chosen, a parity bit is inserted to make the number of high bits (data + parity) even. If odd parity is chosen, the parity bit makes the total number of high bits odd.

Table 22.3. EUART Parity Bits

EUARTn_FRAMECFG_PARITY [1:0]	Description
00	No parity bit (Default)
01	Reserved
10	Even parity
11	Odd parity

22.3.4 Clock Generation

The EUART clock defines the transmission and reception data rate. The baud rate (bit-rate) is given by Figure 22.3 EUART Baud Rate on page 634.

$$br = f_{EUARTn}/(oversample \times (1 + EUARTn\_CLKDIV/256))$$

Figure 22.3. EUART Baud Rate

where f_{EUARTn} is the peripheral clock frequency and oversample is the oversampling rate as defined by OVS in EUARTn_CFG0, see Table 22.4 EUART Oversampling on page 634.

Table 22.4. EUART Oversampling

EUARTn_CFG0_OVS [2:0]	oversample
000	16 (HF mode only)
001	8 (HF mode only)
010	6 (HF mode only)
011	4 (HF mode only)
100	1 (OVS disabled - LF mode only)

Note: In LF mode, with oversampling disabled, the baud rate must be less than 1/3 the LF clock frequency.

The EUART has a fractional clock divider to allow the EUART clock to be controlled more accurately than what is possible with a standard integral divider. The clock divider used in the EUART is a 20-bit value, with a 15-bit integral part and an 5-bit fractional part. The fractional part is configured in the lower 5 bits of DIV in EUARTn_CLKDIV. Fractional clock division is implemented by distributing the selected fraction over thirty two baud periods. The fractional part of the divider tells how many of these periods should be extended by one peripheral clock cycle.

Given a desired baud rate *brdesired*, the clock divider EUARTn_CLKDIV can be calculated by using Figure 22.4 EUART Desired Baud Rate on page 634:

$$EUARTn\_CLKDIV = 256 \times (f_{EUARTn}/(oversample \times brdesired) - 1)$$

Figure 22.4. EUART Desired Baud Rate

Table 22.5 EUART Baud Rates in HF mode @ 4MHz Peripheral Clock with 20 bit CLKDIV on page 634 shows a set of desired baud rates and how accurately the EUART is able to generate these baud rates when running at a 4 MHz peripheral clock in HF mode, using 16x or 8x oversampling.

Table 22.5. EUART Baud Rates in HF mode @ 4MHz Peripheral Clock with 20 bit CLKDIV

Desired baud rate [baud/s]	EUARTn_OVS =00			EUARTn_OVS =01		
	EUARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %	EUARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %
600	415.6563	600.015	0.003	832.3438	599.9925	-0.001
1200	207.3438	1199.94	-0.005	415.6563	1200.03	0.003
2400	103.1563	2400.24	0.010	207.3438	2399.88	-0.005
4800	51.09375	4799.04	-0.020	103.1563	4800.48	0.010
9600	25.03125	9603.842	0.040	51.09375	9598.08	-0.020
14400	16.375	14388.49	-0.080	33.71875	14401.44	0.010
19200	12.03125	19184.65	-0.080	25.03125	19207.68	0.040

Desired baud rate [baud/s]	EUARTn_OVS =00			EUARTn_OVS =01		
	EUARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %	EUARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %
28800	7.6875	28776.98	-0.080	16.375	28776.98	-0.080
38400	5.5	38461.54	0.160	12.03125	38369.3	-0.080
57600	3.34375	57553.96	-0.080	7.6875	57553.96	-0.080
76800	2.25	76923.08	0.160	5.5	76923.08	0.160
115200	1.15625	115942	0.644	3.34375	115107.9	-0.080
230400	0.09375	228571.4	-0.794	1.15625	231884.1	0.644

Table 22.6 EUART Baud Rates in LF mode @ 32.768 kHz Peripheral Clock with 20 bit CLKDIV on page 635 shows a set of desired baud rates and how accurately the EUART is able to generate these baud rates when running from a 32.768 kHz peripheral clock in LF mode.

Table 22.6. EUART Baud Rates in LF mode @ 32.768 kHz Peripheral Clock with 20 bit CLKDIV

Desired baud rate [baud/s]	EUARTn_CLKDIV/256 (to 32nd position)	Actual baud rate [baud/s]	Error %
300	108.21875	300.0217	-0.01
600	53.625	599.8719	0.02
1200	26.3125	1199.744	0.02
2400	12.65625	2399.487	0.02
4800	5.8125	4809.982	-0.21
9600	2.40625	9619.963	-0.21

22.3.5 Auto Baud Detection

The EUART has an automatic baud detection feature, which is available when operating in HF mode. Setting AUTOBAUDEN in EUARTn_CFG0 uses the first frame received to automatically set the baud rate provided that it contains 0x55 (IrDA uses 0x00). The receiver will measure the number of local clock cycles between the beginning of the START bit and the beginning of the 8th data bit. The DIV field in EUARTn_CLKDIV will be overwritten with the new value. The OVS in EUARTn_CFG0 and the +1 count of the Baud Rate equation are already factored into the result that gets written into the DIV field. To restart autobaud detection, clear AUTOBAUDEN and set it high again. Since the auto baud detection is done over 8 baud times, only the upper 3 bits of the fractional part of the clock divider are populated. When autobaud detection has completed, the status bit EUARTn_STATUS_AUTOBAUDDONE and interrupt flag EUARTn_IF_AUTOBAUDDONE are set.

Note:

- If autobaud detection is enabled, software must wait for autobaud detection to complete before transmitting any data.
- Autobaud should be used only during times when it is known that the transmitter will be sending the required data word.
- Autobaud detection is not available in LF mode.
- For autobaud to work with IrDA, there should be odd parity or no parity in the received data frame.

22.3.6 Data Transmission

Asynchronous data transmission is initiated by writing data to the transmit buffer using one of the methods described in 22.3.7 Transmit FIFO Operation. When the transmission shift register is empty and ready for new data, a frame from the transmit FIFO is loaded into the shift register, and if the transmitter is enabled, transmission begins. When the frame has been transmitted, a new frame is loaded into the shift register if available, and transmission continues. If the transmit FIFO is empty, the transmitter goes to an idle state, waiting for a new frame to become available.

Transmission is enabled through the command register EUARTn_CMD by setting TXEN, and disabled by setting TXDIS in the same command register. When the transmitter is disabled using TXDIS, any ongoing transmission is aborted, and any frame currently being transmitted is discarded. When disabled, the TX output goes to an idle state, which by default is a high value. Whether or not the transmitter is enabled at a given time can be read from TXENS in EUARTn_STATUS.

When the EUART transmitter is enabled and there is no data in the transmit shift register or transmit FIFO, the TXC flag in EUARTn_STATUS and the TXC interrupt flag in EUARTn_IF are set, signaling that the transmission is complete. The TXC status flag is cleared when a new frame becomes available for transmission, but the TXC interrupt flag must be cleared by software.

22.3.7 Transmit FIFO Operation

The transmit FIFO is a four entry FIFO buffer. A frame can be loaded into the FIFO by writing to EUARTn_TXDATA. Using EUARTn_TXDATA allows up to 9 bits to be written to the buffer, as well as a set of control bits for the transmission of the written frame. Every frame in the FIFO is stored with 9 data bits and the additional control bits. A frame is loaded from the FIFO into the shift register if the transmitter is enabled.

Figure 22.5 EUART Transmit FIFO Operation on page 637 shows the basics of the transmit FIFO.

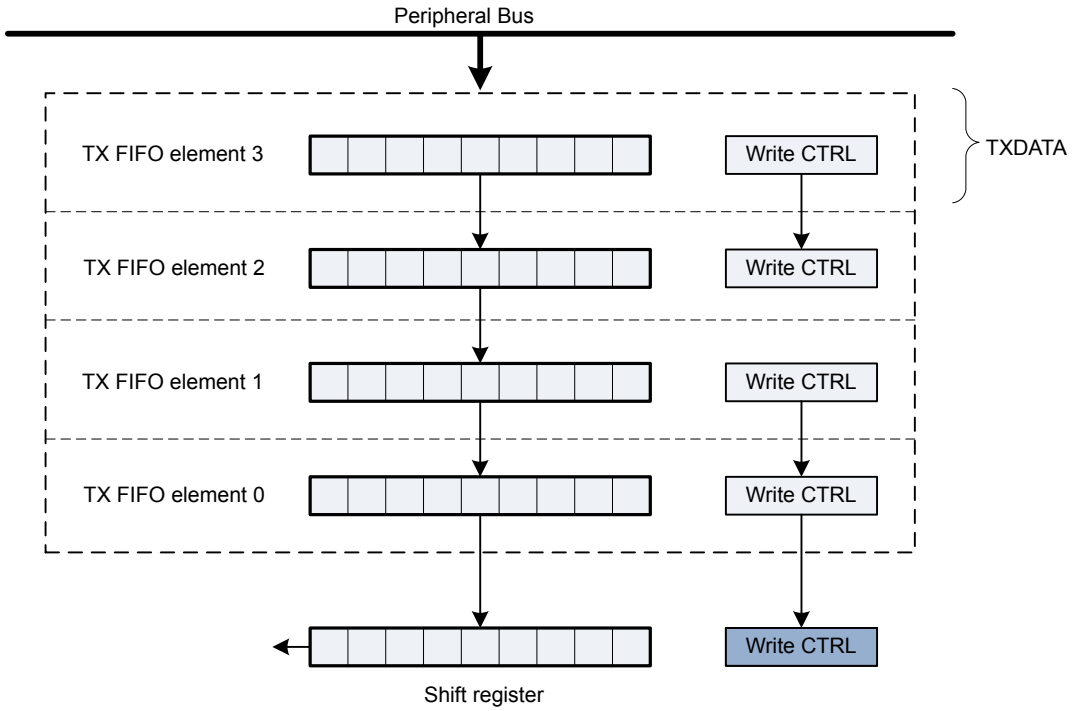


Figure 22.5. EUART Transmit FIFO Operation

In addition to the interrupt flag TXC in EUARTn_IF and status flag TXC in EUARTn_STATUS, which are set when the transmission is complete, TXFL in EUARTn_STATUS and the TXFL interrupt flag in EUARTn_IF are used to indicate the level of the transmit FIFO. The TXFIW field in EUARTn_CFG1 controls the level at which the TXFL flags are set. For example, if TXFIW is set to ONEFRAME, TXFL will be set when the FIFO has space for at least one frame. The TXFCNT field in EUARTn_STATUS indicates the number of entries present in the TX FIFO. If the FIFO is empty, the TXIDLE bit in EUARTn_STATUS will be set to indicate that the transmitter is idle.

The transmit FIFO can be cleared by setting CLEARTX in EUARTn_CMD. Since this is an asynchronous FIFO, software must first issue the CLEARTX command and then wait on the CLEARTXBUSY status flag until it goes low. The EUART must be enabled (EUARTn_EN should be set) for the flush command to work. The EUART should not be transmitting when the CLEARTX command gets issued (can be achieved by disabling the trasmitter). Any frame present in the transmit shift register currently being transmitted will not be aborted due to the flush, and will complete transmission. Note that the transmit shift register is never used to store a transmit frame, i.e., if the transmitter is not enabled then the data stays in the transmit FIFO until the transmitter gets enabled. Whenever a frame is loaded in to the transmit shift register, the transmission starts immediately.

When writing more frames to the transmit buffer than there is free space for, the TXOF interrupt flag in EUARTn_IF will be set, indicating an overflow. The data already in the transmit buffer is preserved in this case, and no data is written.

The FIFO status fields TXFL and TXFCNT are dependent on hardware conditions, and persist even if the EUART is disabled (EUARTn_EN is 0). TXFL will be automatically cleared by hardware if the underlying condition is not met, and TXFCNT always represents the number of words present in the FIFO. Writing to the TX FIFO when the EUART is disabled is allowed.

Note: In LF mode, the TXFL interrupt flag and the TXFL wakeup flag differ slightly in their behavior. IF.TXFL is always set out of reset since there is space available in the FIFO. However, the TXFL wakeup flag is only set after a FIFO read happens and the space that becomes available in the FIFO is the same as programmed in TXFIW in EUARTn_CFG1.

22.3.8 Frame Transmission Control

The transmission control bits, which can be written using EUARTn_TXDATA, affect the transmission of the written frame. The following options are available:

- Generate break: By setting TXBREAK, the output will be held low during the stop-bit period to generate a framing error. A receiver that supports break detection detects this state, allowing it to be used e.g. for framing of larger data packets. The line is driven high before the next frame is transmitted so the next start condition can be identified correctly by the recipient. Continuous breaks lasting longer than a UART frame are thus not supported by the EUART. Direct GPIO control can be used for this.
- Disable transmitter after transmission: If TXDISAT is set, the transmitter is disabled after the frame has been fully transmitted.
- Enable receiver after transmission: If RXENAT is set, the receiver is enabled after the frame has been fully transmitted. It is enabled in time to detect a start-bit directly after the last stop-bit has been transmitted.
- Unblock receiver after transmission: If UBRXAT is set, the receiver is unblocked and RXBLOCK is cleared after the frame has been fully transmitted. See 22.3.12 Blocking Incoming Data for more details.
- Tristate transmitter after transmission: If TXTRIAT is set, TXTRI is set after the frame has been fully transmitted, tristating the transmitter output. Note that if there are more frames in the TX FIFO after the tristating has happened and the transmitter is enabled, then the transmitter will still attempt to send the additional data. Because the output is tristated, nothing will appear at the transmitter output during this time. The FIFO however will get emptied because of the transmitter attempting to send these frames out. If the target is to automatically tristate the TX line whenever the transmitter is idle, then that can be done by setting AUTOTRI in EUARTn_CFG0. If AUTOTRI is set the TXTRI status flag will always read as 0.

22.3.9 Transmission Delay

By configuring TXDELAY in EUARTn_CFG1, the transmitter can be forced to wait a specified number of bit-periods before it transmits the first frame in the TX FIFO. This delay is only applied to the first frame transmitted after the transmitter has been idle. When transmitting frames back-to-back the delay is not introduced between the transmitted frames.

This is useful on half duplex buses, because the receiver always returns received frames to software during the first stop-bit. The bus may still be driven for up to 3 bit periods, depending on the current frame format. Using the transmission delay, a transmission can be started when a frame is received, and it is possible to make sure that the transmitter does not begin driving the output before the frame on the bus is completely transmitted.

22.3.10 Data Reception

Data reception is enabled by setting RXEN bit in EUARTn_CMD. When the receiver is enabled, it actively samples the input looking for a start bit of a new frame. When a start bit is found, reception of the new frame begins if the receive shift register is empty and ready for new data. When the frame has been received, it is pushed into the receive FIFO, making the shift register ready for another frame of data, and the receiver starts looking for another start bit. If a frame is received when the receive FIFO is full, the received frame is discarded and the RXOF interrupt flag in EUARTn_IF is set to indicate a receive FIFO overflow.

The receiver can be disabled by setting the command bit RXDIS in EUARTn_CMD. Any frame currently being received when the receiver is disabled is discarded. The RXENS bit in EUARTn_STATUS indicates whether or not the receiver is enabled.

22.3.11 Receive FIFO Operation

The receive FIFO is a four entry FIFO buffer. Data can be read from the receive FIFO via EUARTn_RXDATA. EUARTn_RXDATA gives access to the received frame. This register also contains parity error and framing error information of the received frame. When a frame is read from the receive FIFO using EUARTn_RXDATA, the frame is pulled out of the FIFO, making room for a new frame. If an attempt is made to read the receive FIFO when it is empty, the RXUF interrupt flag in EUARTn_IF is set to signal a receive buffer underflow. The data returned from a FIFO underflow read is undefined.

Frames can be read from the receive FIFO without removing the data by using EUARTn_RXDATAP. EUARTn_RXDATAP gives access to the first frame in the FIFO, as well as the status bits. The data read from this register when the receive FIFO is empty is undefined. No underflow interrupt is generated by reading using this register, i.e. RXUF in EUARTn_IF is never set as a result of reading from EUARTn_RXDATAP.

The basic operation of the receive FIFO is shown in Figure 22.6 EUART Receive FIFO Operation on page 639.

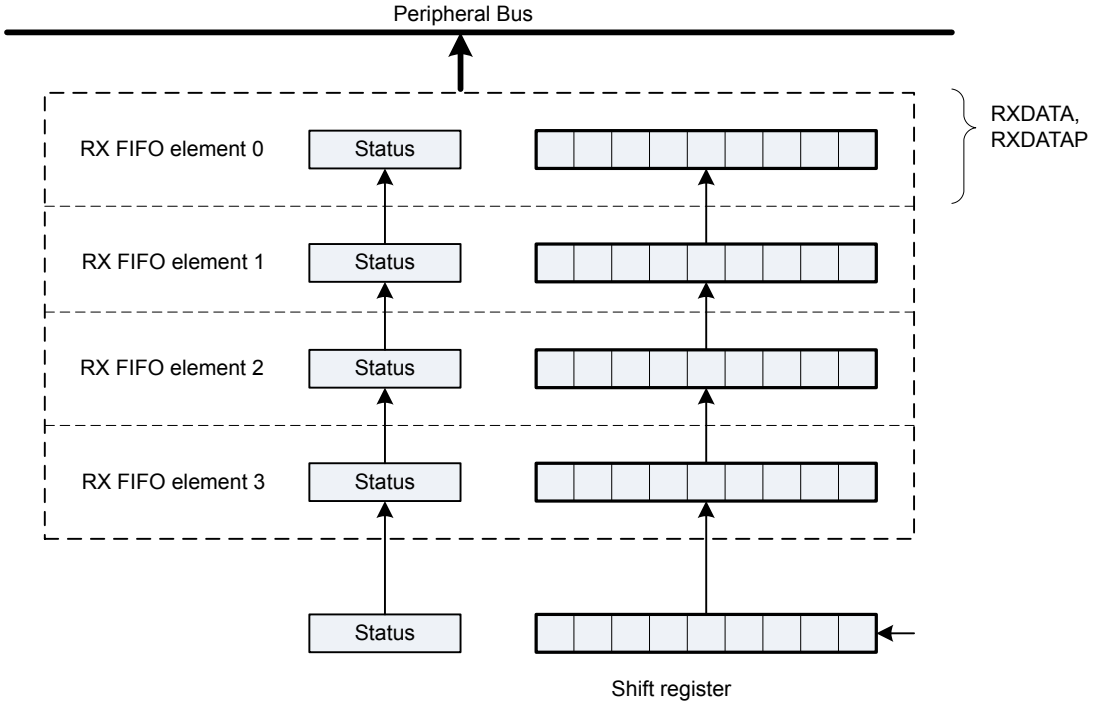


Figure 22.6. EUART Receive FIFO Operation

The receive FIFO has two associated status flags: RXFL (set when number of available frames in the receive FIFO is at least number of frames set by RXFIW in the CFG1 register) and RXFULL (set when receive FIFO is full). These status flags remain set as long as the underlying condition is true, even if the EUART is disabled (i.e., EUARTn_EN is 0). It is possible to read from the receive FIFO while EUART is disabled, this will impact the two status flags mentioned above. The status flags RXFL and RXFULL are automatically cleared by hardware when their condition is no longer true.

The receive FIFO has four associated interrupt flags in the IF register: RXFL, RXFULL, RXOF and RXUF. RXFL is set when number of available frames in the receive FIFO is at least number of frames set by RXFIW in the CFG1 register. RXFULL is set when the receive FIFO is full. Both RXFL and RXFULL remain set as long as the underlying condition is true even if the EUART is disabled. This means that if a software clear is done for RXFL / RXFULL while the undelying condition of respective interrupt is still true, the corresponding interrupt will get set again after the clear (and this will happen even if EUART is disabled). Reading data from the FIFO or disabling the RXFL / RXFULL interrupts will block the respective interrupt after a software clear. RXOF is set when a new frame is received while the receive FIFO is full, indicating a receive FIFO overflow. The new frame is discarded. RXOF triggers every time an overflow event occurs. RXUF (receive FIFO underflow) is set when an attempt is made by software or DMA to read the receive FIFO when it is empty. The data read from the FIFO is undefined. RXUF triggers every time an underflow occurs even if the EUART is disabled.

22.3.12 Blocking Incoming Data

When using hardware frame recognition, as detailed in 22.3.23 Multi-Processor Mode and 22.3.24 Collision Detection, it is necessary to be able to let the receiver sample incoming frames without pushing them into the receive FIFO. This is accomplished by blocking incoming data.

Incoming data is blocked as long as RXBLOCK in EUARTn_STATUS is set. When blocked, frames received by the receiver will not be loaded into the receive FIFO, and software is not notified by the RXFL flag in EUARTn_STATUS or the RXFL interrupt flag in EUARTn_IF of their arrival. For data to be loaded into the receive buffer, RXBLOCK must be cleared before a frame is fully received by the receiver. RXBLOCK is set by setting RXBLOCKEN in EUARTn_CMD and disabled by setting RXBLOCKDIS also in EUARTn_CMD. There is one exception where data is loaded into the receive FIFO even when RXBLOCK is set. This is when an address frame is received when operating in multi-processor mode. See 22.3.23 Multi-Processor Mode for more information.

Frames received containing framing or parity errors will not result in the FERR and PERR interrupt flags in EUARTn_IF being set while RXBLOCK in EUARTn_STATUS is set. Hardware recognition is not applied to these erroneous frames, and they are silently discarded.

22.3.13 Data Sampling and Filtering

The receiver samples the incoming signal at a rate 16, 8, 6 or 4 times higher than the given baud rate, depending on the oversampling mode given by OVS in EUARTn_CFG0. Lower oversampling rates make higher baud rates possible, but give less room for errors.

When a high-to-low transition is registered on the input while the receiver is idle, this is recognized as a start-bit, and the baud rate generator is synchronized with the incoming frame.

For oversampling modes 16, 8 and 6, every bit in the incoming frame is sampled three times to gain a level of noise immunity. These samples are aimed at the middle of the bit-periods, as visualized in Figure 22.7 EUART Sampling of Start and Data Bits on page 641. With OVS=0 in EUARTn_CFG0, the start and data bits are thus sampled at locations 8, 9 and 10 in the figure, locations 4, 5 and 6 for OVS=1 and locations 3, 4, and 5 for OVS=2. The value of a sampled bit is determined by majority vote. If two or more of the three bit-samples are high, the resulting bit value is high. If the majority is low, the resulting bit value is low.

Majority vote is used for all oversampling modes except 4x oversampling and when oversampling is disabled. In 4x oversampling mode, a single sample is taken at position 3 as shown in Figure 22.7 EUART Sampling of Start and Data Bits on page 641.

When oversampling is disabled i.e. OVS = DISABLE, there is only one available location for sampling the start and data bits and so majority vote is not used.

Note: When operating in HF mode, oversampling must be set to 4, 6, 8, or 16x. When operating in LF mode, oversampling must be disabled.

Software can disable the majority vote behavior by setting MVDIS in EUARTn_CFG0. When majority vote is disabled by software, a single sample is taken at location 9 in the figure for OVS=0, location 5 for OVS=1 and location 4 for OVS=2.

If the value of the start bit is found to be high, the reception of the frame is aborted, filtering out false start bits possibly generated by noise on the input.

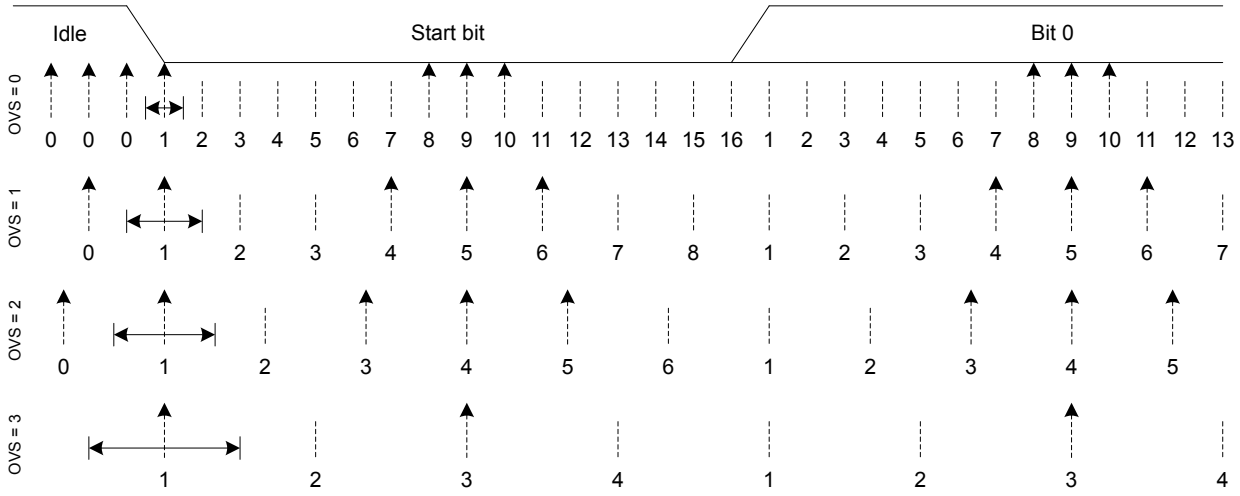


Figure 22.7. EUART Sampling of Start and Data Bits

If the baud rate of the transmitter and receiver differ, the location each bit is sampled will be shifted towards the previous or next bit in the frame. This is acceptable for small errors in the baud rate, but for larger errors, it will result in transmission errors.

When the number of stop bits is 1 or more, stop bits are sampled like the start and data bits as seen in Figure 22.8 EUART Sampling of Stop Bits when Number of Stop Bits are 1 or More on page 642. When a stop bit has been detected, the EUART is ready for a new start bit.

As seen in Figure 22.8 EUART Sampling of Stop Bits when Number of Stop Bits are 1 or More on page 642, a stop-bit of length 1 normally ends at c, but the next frame will be received correctly as long as the start-bit comes after position a for OVS=0 and OVS=3, and b for OVS=1 and OVS=2.

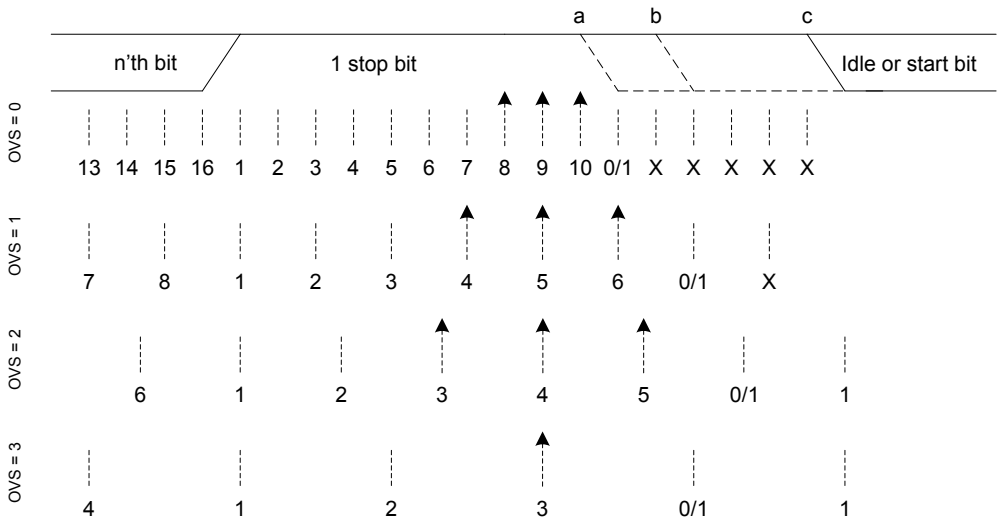


Figure 22.8. EUART Sampling of Stop Bits when Number of Stop Bits are 1 or More

When working with stop bit lengths of half a baud period, the above sampling scheme no longer suffices. In this case, the stop-bit is not sampled, and no framing error is generated in the receiver if the stop-bit is not generated. The line must still be driven high before the next start bit however for the EUART to successfully identify the start bit.

22.3.14 Parity Error

When parity bits are enabled, a parity check is automatically performed on incoming frames. When a parity error is detected in an incoming frame, the data parity error bit PERR in the frame is set, as well as the interrupt flag PERR in EUARTn_IF. Frames with parity errors are loaded into the receive FIFO like regular frames.

PERR can be accessed by reading the frame from the receive FIFO using the EUARTn_RXDATA or EUARTn_RXDATAXP registers.

If ERRSTX in EUARTn_CFG0 is set, the transmitter is disabled on received parity errors. If ERRSRX in EUARTn_CFG0 is set, the receiver is disabled on received parity errors.

22.3.15 Framing Error and Break Detection

A framing error is the result of an asynchronous frame where the stop bit was sampled to a value of 0. This can be the result of noise and baud rate errors, but can also be the result of a break generated by the transmitter on purpose.

When a framing error is detected in an incoming frame, the framing error bit FERR in the frame is set. The interrupt flag FERR in EUARTn_IF is also set. Frames with framing errors are loaded into the receive FIFO like regular frames.

FERR can be accessed by reading the frame from the receive buffer using the EUARTn_RXDATAX or EUARTn_RXDATAXP registers.

If ERRSTX in EUARTn_CFG0 is set, the transmitter is disabled on received framing errors. If ERRSRX in EUARTn_CFG0 is set, the receiver is disabled on received framing errors.

22.3.16 Programmable Start Frame

The EUART can be configured to start receiving data when a special start frame is detected on the input. This can be useful when operating in low energy modes, allowing other devices to gain the attention of the EUART by transmitting a given frame.

When SFUBRX in EUARTn_CFG1 is set, an incoming frame matching the frame defined in EUARTn_STARTFRAME will result in RXBLOCK in EUARTn_STATUS being cleared. This can be used to enable reception when a specified start frame is detected. If the receiver is enabled and blocked, i.e. RXENS and RXBLOCK in EUARTn_STATUS are set, the receiver will receive all incoming frames, but unless an incoming frame is a start frame it will be discarded and not loaded into the receive FIFO. When a start frame is detected, the block is cleared, and frames received from that point, including the start frame, are loaded into the receive FIFO.

An incoming start frame results in the STARTFIF interrupt flag in EUARTn_IF being set, regardless of the value of SFUBRX in EUARTn_CFG1. This allows an interrupt to be made when the start frame is detected. The interrupt will be set even if the receiver is blocked i.e. EUARTn_STATUS_RXBLOCK = 1.

Note: The receiver must be enabled for start frames to be detected. Please note that, if another UART device sends a start frame but a parity and/or framing error occurs during the receiption, the received frame is not detected as a start frame.

22.3.17 Programmable Signal Frame

As well as the configurable start frame, a special signal frame can be specified. When a frame matching the frame defined in EUARTn_SIGFRAME is detected by the receiver, the SIGFIF interrupt flag in EUARTn_IF is set. As with start frame detection, the interrupt will be set even if the receiver is blocked i.e. EUARTn_STATUS_RXBLOCK = 1.

One use of the programmable signal frame is to signal the end of a multi-frame message transmitted to the EUART. An interrupt will then be triggered when the packet has been completely received, allowing software to process it. Used in conjunction with the programmable start frame and DMA, this makes it possible for the EUART to automatically begin the reception of a packet on a specified start frame, load the entire packet into memory, and give an interrupt when reception of a packet has completed. When one of the low frequency oscillators (LFXO, LFRCO, ULFRCO) is used as the EUARTn peripheral clock source, the device can thus wait for data packets in EM2, and only be woken up when a packet has been completely received.

Note: The receiver must be enabled for a signal frame to be detected. If a parity and/or framing error occurs during the reception of a signal frame, the received frame is not detected as a signal frame.

22.3.18 Local Loopback

The EUART receiver samples RX by default, and the transmitter drives TX by default. This is not the only option however. When LOOPBK in EUARTn_CFG0 is set, the receiver is connected to the TX pin as shown in Figure 22.9 EUART Local Loopback on page 643. This is useful for debugging, as the EUART can receive the data it transmits, but it is also used to allow the EUART to read and write to the same pin, which is required for some half duplex communication modes. In this mode, the TX pin must be enabled as an output in the GPIO.

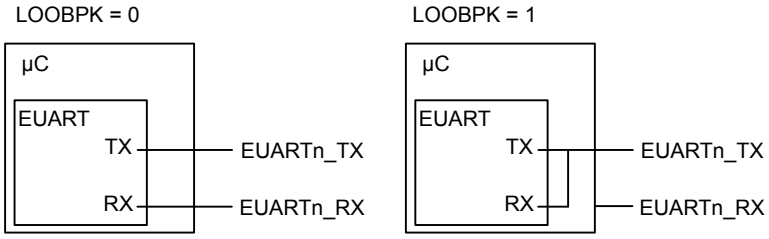


Figure 22.9. EUART Local Loopback

22.3.19 Half Duplex Communication

When doing full duplex communication, two data links are provided, making it possible for data to be sent and received at the same time. In half duplex mode, data is only sent in one direction at a time. There are several possible half duplex setups, as described in the following sections.

22.3.20 Single Data-link

In this setup, the EUART both receives and transmits data on the same pin. This is enabled by setting LOOPBK in EUARTn_CFG0, which connects the receiver to the transmitter output. Because they are both connected to the same line, it is important that the EUART transmitter does not drive the line when receiving data, as this would corrupt the data on the line.

When communicating over a single data-link, the transmitter must thus be tristated whenever not transmitting data. This is done by setting the command bit TXTRIEN in EUARTn_CMD, which tristates the transmitter. Before transmitting data, the command bit TXTRIDIS, also in EUARTn_CMD, must be set to enable transmitter output again. Whether or not the output is tristated at a given time can be read from TXTRI in EUARTn_STATUS. If TXTRI is set when transmitting data, the data is shifted out of the shift register, but is not put out on the TX line.

When operating a half duplex data bus, it is common to have a bus master, which first transmits a request to one of the bus slaves, then receives a reply. In this case, the frame transmission control bits, which can be set by writing to EUARTn_TXDATA, can be used to make the EUART automatically disable transmission, tristate the transmitter and enable reception when the request has been transmitted, making it ready to receive a response from the slave.

The transmission delay feature, detailed in 22.3.9 Transmission Delay, can also be used to add delay between the RX and TX frames so that the interrupt service routine has time to process data that was just received before transmitting more data. Hardware flow control is another method to insert time for processing the frame. RTS and CTS can be used to halt either the link partner's transmitter or the local transmitter. See the section on hardware flow control, 22.3.25 Hardware Flow Control, for more details.

Tristating the transmitter can also be performed automatically by the EUART by using AUTOTRI in EUARTn_CFG0. When AUTOTRI is set, the EUART automatically tristates U(S)n_TX whenever the transmitter is idle, and enables transmitter output when the transmitter goes active. If AUTOTRI is set TXTRI is always read as 0.

Note: Another way to tristate the transmitter is to enable wired-and or wired-or mode in GPIO. For wired-and mode, outputting a 1 will be the same as tristating the output, and for wired-or mode, outputting a 0 will be the same as tristating the output. This can only be done on buses with a pull-up or pull-down resistor, respectively.

22.3.21 Single Data-link with External Driver

Some communication schemes, such as RS-485 rely on an external driver. Here, the driver has an extra input which enables it, and instead of tristating the transmitter when receiving data, the external driver must be disabled.

This can be done manually by assigning a GPIO to turn the driver on or off.

Figure 22.10 EUART Half Duplex Communication with External Driver on page 644 shows an example configuration using an external driver.

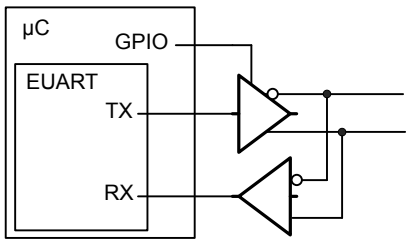


Figure 22.10. EUART Half Duplex Communication with External Driver

22.3.22 Two Data-links

Some limited devices only support half duplex communication even though two data links are available. In this case software is responsible for making sure data is not transmitted when incoming data is expected.

22.3.23 Multi-Processor Mode

To simplify communication between multiple processors, the EUART supports a special multi-processor mode. In this mode the 9th data bit in each frame is used to indicate whether the content of the remaining 8 bits is data or an address.

When multi-processor mode is enabled, an incoming 9-bit frame with the 9th bit equal to the value of MPAB in EUARTn_CFG0 is identified as an address frame. When an address frame is detected, the MPAF interrupt flag in EUARTn_IF is set, and the address frame is loaded into the receive register. This happens regardless of the value of RXBLOCK in EUARTn_STATUS.

Multi-processor mode is enabled by setting MPM in EUARTn_CFG0, and the value of the 9th bit in address frames can be set in EUARTn_CFG0_MPAB. Note that the receiver must be enabled for address frames to be detected. The receiver can be blocked however, preventing data from being loaded into the receive FIFO while looking for address frames.

When a slave has received an address frame and wants to receive the following data, it must make sure the receiver is unblocked before the next frame has been completely received in order to prevent data loss.

22.3.23.1 EUART Multi-Processor Mode Example

1. All slaves enable multi-processor mode and, enable and block the receiver. They will now not receive data unless it is an address frame. MPAB in EUARTn_CFG0 is set to identify frames with the 9th bit equal to MPAB as address frames.
2. The master sends a frame containing the address of a slave and with the 9th bit set to the value of MPAB.
3. All slaves receive the address frame and get an interrupt. They can read the address from the receive FIFO. The selected slave unblocks the receiver to start receiving data from the master.
4. The master sends data with the 9th bit set to the opposite value of MPAB.
5. Only the slave with RX enabled and unblocked receives the data. When transmission is complete, the slave blocks the receiver and waits for a new address frame.

22.3.24 Collision Detection

The EUART supports a basic form of collision detection. When the receiver is connected to the output of the transmitter, either by using the LOOPBK bit in EUARTn_CFG0 or through an external connection, this feature can be used to detect whether data transmitted on the bus by the EUART did get corrupted by a simultaneous transmission by another device on the bus.

For collision detection to be enabled, CCEN in EUARTn_CFG0 must be set, and the receiver enabled. The data sampled by the receiver is then continuously compared with the data output by the transmitter. If they differ, the CCF interrupt flag in EUARTn_IF is set. The collision check includes all bits of the transmitted frames. The CCF interrupt flag is set once for each bit sampled by the receiver that differs from the bit output by the transmitter. When the transmitter output is disabled, i.e. the transmitter is tristated, collisions are not registered.

Note: Collision detection is only supported for baud rates up to 1 Mbps.

22.3.25 Hardware Flow Control

Hardware flow control can be used to hold off the link partner's transmission until RX buffer space is available. Port and pin selection for RTS and CTS are configured in GPIO_EUARTn_RTSROUTE and GPIO_EUARTn_CTSROUTE. RTSPEN in GPIO_EUARTn_ROUTEEN enables the RTS pin as an output.

RTS is an output signal which indicates that RX FIFO space is available to receive a frame. The link partner is being requested to send its data when RTS is asserted. RTS activation can be made dependent on how much space is available in the receive FIFO using RTXRXFW in EUARTn_CFG1. For debug use set DBGHALT in EUARTn_CFG1 which will force the RTS to request one frame from the link partner when the CPU core single steps. RTS is deactivated when RX is disabled.

CTS is an input signal that is used to halt transmission. The link partner controls CTS to indicate whether ot not there is space in its receive FIFO to accept new data. It will assert CTS when there is space available, and deassert CTS if there is no more room. If CTS deactivates in the middle of a frame, the frame currently being transmitted is completed before stopping. CTS operation needs to be enabled using EUARTn_CFG1_CTSEN.

The RTS and CTS signals are active low by default, but their polarity can be changed with RTSINV and CTSINV in EUARTn_CFG1.

22.3.26 Debug Halt

During debugging, the debug halt feature allows halting EUART reception by deactivating RTS when the core is halted and continuing frame reception by activating RTS when the core is released. EUART debug halt can be enabled by setting DBGHALT in EUSART_CFG1 to '1'. The EUART receiver must be enabled for debug halting to function.

When EUSART_CFG1_DBGHALT is not set or EUSART_CFG1_DBGHALT is set but the debugger has not halted the system, RTS is only dependent on the receive FIFO having space available to receive the specified number of frames given by EUSART_CFG1_RTSRXFW.

When EUSART_CFG1_DBGHALT is set, RTS will remain deactivated as long as the debugger has halted the system. When the debugger releases the system temporarily while DBGHALT is set, RTS will be activated if the receive FIFO has space available to receive at least EUSART_CFG1_RTSRXFW frames, and no frame is being received. RTS will be deactivated again when the debugger halt is re-instated, and the receiver starts receiving a new frame or if the receive FIFO does not have space available to receive at least EUSART_CFG1_RTSRXFW frames. This behavior allows single stepping in the debugger to pulse the system halt signal for a cycle, and receive the next frame.

As the incoming frame is always received until the receive FIFO is full regardless of the state of DBGHALT or whether the chip is halted, the link partner must honor the RTS signal and stop transmitting when RTS is deactivated, or the receive FIFO could overflow.

All data in the transmit FIFO is sent out even when a debugger halt is active; therefore, DMA will need to be configured to stop sending EUART TX data during a debug halt.

22.3.27 PRS-triggered Transmissions

If a transmission must be started on an event with very little delay, the PRS system can be used to trigger the transmission. The PRS channel to use as a trigger can be selected using PRSSEL in PRS_EUARTn_TRIGGER. When a positive edge is detected on this signal, the receiver is enabled if RXTEN in EUARTn_TRIGCTRL is set, and the transmitter is enabled if TXTEN in EUARTn_TRIGCTRL is set. Only one signal input is supported by the EUART.

22.3.28 PRS RX Input

The EUART can be configured to receive data directly from a PRS channel by setting RXPRSEN in EUARTn_CFG1. The PRS channel used is selected using PRSSEL in PRS_EUARTn_RX.

22.3.29 DMA Support

The EUART has full DMA support. The DMA controller can write to the transmit FIFO using the EUARTn_TXDATA register, and it can read from the receive FIFO using the register EUARTn_RXDATA. This enables single byte transfers, 9 bit data + control/status bits transfers both to and from the EUART.

A request for the DMA controller to read from the EUART receive buffer can be triggered when the receive FIFO watermark is crossed, i.e. at least RXFIW frames of data are available in the FIFO.

A write request for the DMA controller can be triggered when the transmit FIFO watermark is crossed, i.e. at least TXFIW frames in the transmit buffer are empty.

In some cases, it may be sensible to temporarily stop DMA access to the EUART when an error such as a framing error has occurred. This is enabled by setting ERRSDMA in EUARTn_CFG0.

The EUART can also work with the DMA when operating in EM2 (in LF mode) so that the system does not have to wake up to EM0 or EM1 to consume data. This functionality is enabled by setting TXDMAWU or RXDMAWU in the EUARTn_CFG1 register. The DMA will be triggered by the EUART when the TX or RX watermark conditions are met. DMA will then temporarily power up and pop/push all the elements of the corresponding FIFO, and then power off again without waking the CPU.

22.3.30 Interrupts

The interrupts generated by the EUART are combined into two interrupt vectors. Interrupts related to reception are assigned to one interrupt vector, and interrupts related to transmission are assigned to the other. Separating the interrupts in this way allows different priorities to be set for transmission and reception interrupts.

The transmission interrupt vector groups the transmission-related interrupts generated by the following interrupt flags:

- TXC
- TXFL
- TXOF
- CCF
- TXIDLE

The reception interrupt on the other hand groups the reception-related interrupts, triggered by the following interrupt flags:

- RXFL
- RXFULL
- RXOF
- RXUF
- PERR
- FERR
- MPAF
- START
- SIG
- AUTOBAUDDONE

If EUART interrupts are enabled, an interrupt will be made if one or more of the interrupt flags in EUARTn_IF and their corresponding bits in EUARTn_IEN are set. In LE mode, all interrupts can serve to wake the system from EM2.

22.3.31 EM2 Operation (LF Mode Only)

The EUART can operate in EM2 when running from an LF oscillator source (LF mode). Note that sending and receiving data in EM2 requires that the EUART be connected to GPIO that are capable of operating in EM2. This includes all pins on Port A and Port B. Pins on Port C and Port D are not available for digital peripheral signalling in EM2 or EM3.

EM2 operation allows the EUART to wait for an incoming UART frame, or even wait on the programmable start or signal frames while the system is consumiung very little energy. When a UART frame is completely received, or a start/signal frame is detected, the CPU can quickly be woken up. Alternatively, multiple frames can be transferred via the Direct Memory Access (DMA) module into RAM memory before waking up the CPU. Similarly, data can be transmitted in EM2 with data from the CPU or through use of the DMA.

All interrupts can be used as wake up interrupts if enabled. None of the interrupts are sticky, i.e., the interrupt triggers only once whenever the interrupt condition is reached.

Note: When RXDMAWU or TXDMAWU is set in EUARTn_CFG1, the system will not be able to go to EM2 before all related EUART DMA requests have been processed. This means that if RXDMAWU is set and the EUART receives a frame, the system will wait to go to EM2 before the frame has been read from the EUART. In order for the system to go to EM2 during the last byte transmission, TXDMAWU must be cleared in the DMA interrupt service routine. This is because TXFL will be high during the final byte transfer.

22.3.32 IrDA Modulator/ Demodulator

The IrDA modulator implements the physical layer of the IrDA specification, which is necessary for communication over IrDA. The modulator takes the signal output from the EUART module, and modulates it before it leaves the EUART. In the same way, the input signal is demodulated before it enters the actual EUART module. The modulator implements the original Rev. 1.0 physical layer and one high speed extension which supports speeds from 2.4 kbps to 1.152 Mbps.

The data from and to the EUART is represented in a NRZ (Non Return to Zero) format, where the signal value is at the same level through the entire bit period. For IrDA, the required format is RZI (Return to Zero Inverted), a format where a “1” is signalled by holding the line low, and a “0” is signalled by a short high pulse. An example is given in Figure 22.11 EUART Example RZI Signal for a given Asynchronous UART Frame on page 648.

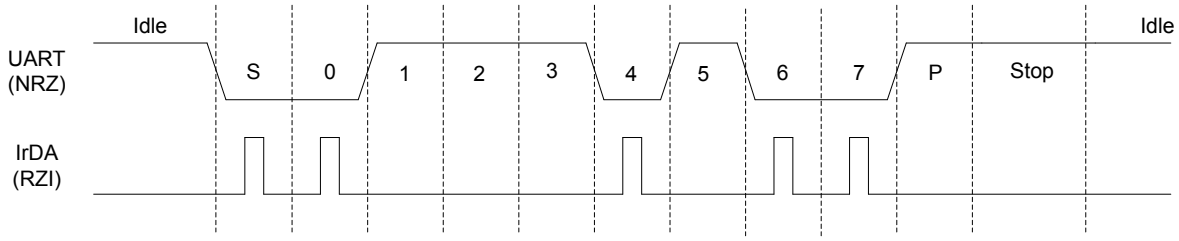


Figure 22.11. EUART Example RZI Signal for a given Asynchronous UART Frame

The IrDA module is enabled by setting IREN. The EUART transmitter output and receiver input is then routed through the IrDA modulator.

The width of the pulses generated by the IrDA modulator is set by configuring IRPW in EUARTn_IRCTRL. Four pulse widths are available, each defined relative to the configured bit period as listed in Table 22.7 EUART IrDA Pulse Widths on page 648.

Table 22.7. EUART IrDA Pulse Widths

IRPW	Pulse width OVS=0	Pulse width OVS=1	Pulse width OVS=2	Pulse width OVS=3
00	1/16	1/8	1/6	1/4
01	2/16	2/8	2/6	N/A
10	3/16	3/8	N/A	N/A
11	4/16	N/A	N/A	N/A

By default, no filter is enabled in the IrDA demodulator. A filter can be enabled by setting IRFILT in EUARTn_IRCTRL. When the filter is enabled, an incoming pulse has to last for 4 consecutive clock cycles to be detected by the IrDA demodulator.

Note that by default, the idle value of the EUART data signal is high. This means that the IrDA modulator generates negative pulses, and the IrDA demodulator expects negative pulses. To make the IrDA module use RZI signalling, both TXINV and RXINV in EUARTn_CFG0 must be set.

22.4 EUSART Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	EUSART_IPVERSION	R	IP version ID
0x004	EUSART_EN	RW ENABLE	Enable Register
0x008	EUSART_CFG0	RW CONFIG	Configuration 0 Register
0x00C	EUSART_CFG1	RW CONFIG	Configuration 1 Register
0x010	EUSART_FRAMECFG	RW CONFIG	Frame Format Register
0x014	EUSART_IRHFCFG	RW CONFIG	HF IrDA Mod Config Register
0x018	EUSART_IRLFCFG	RW CONFIG	LF IrDA Pulse Config Register
0x01C	EUSART_TIMINGCFG	RW CONFIG	Timing Register
0x020	EUSART_STARTFRAMECFG	RW CONFIG	Start Frame Register
0x024	EUSART_SIGFRAMECFG	RW CONFIG	Signal Frame Register
0x028	EUSART_CLKDIV	RWH LFSYNC	Clock Divider Register
0x02C	EUSART_TRIGCTRL	RW LFSYNC	Trigger Control Register
0x030	EUSART_CMD	W LFSYNC	Command Register
0x034	EUSART_RXDATA	RH	RX Data Register
0x038	EUSART_RXDATAP	RH	RX Data Peek Register
0x03C	EUSART_TXDATA	W	TX Data Register
0x040	EUSART_STATUS	RH	Status Register
0x044	EUSART_IF	RWH INTFLAG	Interrupt Flag Register
0x048	EUSART_IEN	RW	Interrupt Enable Register
0x04C	EUSART_SYNCBUSY	RH	Synchronization Busy Register
0x1000	EUSART_IPVERSION_SET	R	IP version ID
0x1004	EUSART_EN_SET	RW ENABLE	Enable Register
0x1008	EUSART_CFG0_SET	RW CONFIG	Configuration 0 Register
0x100C	EUSART_CFG1_SET	RW CONFIG	Configuration 1 Register
0x1010	EUSART_FRAMECFG_SET	RW CONFIG	Frame Format Register
0x1014	EUSART_IRHFCFG_SET	RW CONFIG	HF IrDA Mod Config Register
0x1018	EUSART_IRLFCFG_SET	RW CONFIG	LF IrDA Pulse Config Register
0x101C	EUSART_TIMINGCFG_SET	RW CONFIG	Timing Register
0x1020	EUSART_STARTFRA-MECFG_SET	RW CONFIG	Start Frame Register
0x1024	EUSART_SIGFRAMECFG_SET	RW CONFIG	Signal Frame Register
0x1028	EUSART_CLKDIV_SET	RWH LFSYNC	Clock Divider Register
0x102C	EUSART_TRIGCTRL_SET	RW LFSYNC	Trigger Control Register
0x1030	EUSART_CMD_SET	W LFSYNC	Command Register
0x1034	EUSART_RXDATA_SET	RH	RX Data Register

Offset	Name	Type	Description
0x1038	EUSART_RXDATAP_SET	RH	RX Data Peek Register
0x103C	EUSART_TXDATA_SET	W	TX Data Register
0x1040	EUSART_STATUS_SET	RH	Status Register
0x1044	EUSART_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1048	EUSART_IEN_SET	RW	Interrupt Enable Register
0x104C	EUSART_SYNCBUSY_SET	RH	Synchronization Busy Register
0x2000	EUSART_IPVERSION_CLR	R	IP version ID
0x2004	EUSART_EN_CLR	RW ENABLE	Enable Register
0x2008	EUSART_CFG0_CLR	RW CONFIG	Configuration 0 Register
0x200C	EUSART_CFG1_CLR	RW CONFIG	Configuration 1 Register
0x2010	EUSART_FRAMECFG_CLR	RW CONFIG	Frame Format Register
0x2014	EUSART_IRHFCFG_CLR	RW CONFIG	HF IrDA Mod Config Register
0x2018	EUSART_IRLFCFG_CLR	RW CONFIG	LF IrDA Pulse Config Register
0x201C	EUSART_TIMINGCFG_CLR	RW CONFIG	Timing Register
0x2020	EUSART_STARTFRA-MECFG_CLR	RW CONFIG	Start Frame Register
0x2024	EUSART_SIGFRAMECFG_CLR	RW CONFIG	Signal Frame Register
0x2028	EUSART_CLKDIV_CLR	RWH LFSYNC	Clock Divider Register
0x202C	EUSART_TRIGCTRL_CLR	RW LFSYNC	Trigger Control Register
0x2030	EUSART_CMD_CLR	W LFSYNC	Command Register
0x2034	EUSART_RXDATA_CLR	RH	RX Data Register
0x2038	EUSART_RXDATAP_CLR	RH	RX Data Peek Register
0x203C	EUSART_TXDATA_CLR	W	TX Data Register
0x2040	EUSART_STATUS_CLR	RH	Status Register
0x2044	EUSART_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2048	EUSART_IEN_CLR	RW	Interrupt Enable Register
0x204C	EUSART_SYNCBUSY_CLR	RH	Synchronization Busy Register
0x3000	EUSART_IPVERSION_TGL	R	IP version ID
0x3004	EUSART_EN_TGL	RW ENABLE	Enable Register
0x3008	EUSART_CFG0_TGL	RW CONFIG	Configuration 0 Register
0x300C	EUSART_CFG1_TGL	RW CONFIG	Configuration 1 Register
0x3010	EUSART_FRAMECFG_TGL	RW CONFIG	Frame Format Register
0x3014	EUSART_IRHFCFG_TGL	RW CONFIG	HF IrDA Mod Config Register
0x3018	EUSART_IRLFCFG_TGL	RW CONFIG	LF IrDA Pulse Config Register
0x301C	EUSART_TIMINGCFG_TGL	RW CONFIG	Timing Register
0x3020	EUSART_STARTFRA-MECFG_TGL	RW CONFIG	Start Frame Register
0x3024	EUSART_SIGFRAMECFG_TGL	RW CONFIG	Signal Frame Register

Offset	Name	Type	Description
0x3028	EUSART_CLKDIV_TGL	RWH LFSYNC	Clock Divider Register
0x302C	EUSART_TRIGCTRL_TGL	RW LFSYNC	Trigger Control Register
0x3030	EUSART_CMD_TGL	W LFSYNC	Command Register
0x3034	EUSART_RXDATA_TGL	RH	RX Data Register
0x3038	EUSART_RXDATAP_TGL	RH	RX Data Peek Register
0x303C	EUSART_TXDATA_TGL	W	TX Data Register
0x3040	EUSART_STATUS_TGL	RH	Status Register
0x3044	EUSART_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3048	EUSART_IEN_TGL	RW	Interrupt Enable Register
0x304C	EUSART_SYNCBUSY_TGL	RH	Synchronization Busy Register

22.5 EUSART Register Description

22.5.1 EUSART_IPVERSION - IP version ID

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

22.5.2 EUSART_EN - Enable Register

Offset	Bit Position																														
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**Module enable**
	Set to enable the module.			

22.5.3 EUSART_CFG0 - Configuration 0 Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0						0x0	0x0	0x0		0x0			0x0			0x0	0x0			0x0			0x0			0x0	0x0	0x0	0x0	
Access	RW	RW						RW	RW	RW		RW			RW			RW	RW			RW			RW			RW	RW	RW	RW	
Name	AUTOBAUDEN	MVDIS						ERRSTX	ERRSRX	ERRSDMA		SKIPPERRF			AUTOTRI			TXINV	RXINV			MSBF			OVS			MPAB	MPM	CCEN	LOOPBK	

Bit	Name	Reset	Access	Description
31	AUTOBAUDEN	0x0	RW	**AUTOBAUD detection enable**
	Detects the baud rate based on receiving a 0x55 frame (0x00 for IrDA).			
30	MVDIS	0x0	RW	**Majority Vote Disable**
	Disable majority vote for 16x, 8x and 6x oversampling modes.			
29:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	ERRSTX	0x0	RW	**Disable TX On Error**
	When set, the transmitter is disabled on framing and parity errors in the receiver.			

Value	Mode	Description
0	DISABLE	Received framing and parity errors have no effect on transmitter
1	ENABLE	Received framing and parity errors disable the transmitter

Bit	Name	Reset	Access	Description
23	ERRSRX	0x0	RW	**Disable RX On Error**
	When set, the receiver is disabled on framing and parity errors.			

Value	Mode	Description
0	DISABLE	Framing and parity errors have no effect on receiver
1	ENABLE	Framing and parity errors disable the receiver

Bit	Name	Reset	Access	Description
22	ERRSDMA	0x0	RW	**Halt DMA Read On Error**
	When set, DMA read requests will be cleared on framing and parity errors.			

Value	Mode	Description
0	DISABLE	Framing and parity errors have no effect on DMA requests from the UART
1	ENABLE	DMA requests from the UART are blocked while the PERR or FERR interrupt flags are set

Bit	Name	Reset	Access	Description
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	SKIPPERRF	0x0	RW	**Skip Parity Error Frames**
	When set, the receiver discards frames with parity errors. The PERR interrupt flag is still set.			

Bit	Name	Reset	Access	Description
19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	AUTOTRI	0x0	RW	**Automatic TX Tristate**

When enabled, TXTRI is set by hardware whenever the transmitter is idle, and TXTRI is cleared by hardware when transmission starts.

Value	Mode	Description
0	DISABLE	The output on UARTn_TX when the transmitter is idle is defined by TXINV
1	ENABLE	UARTn_TX is tristated whenever the transmitter is idle

16:15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	TXINV	0x0	RW	**Transmitter output Invert**

The output from the EUART transmitter can optionally be inverted by setting this bit.

Value	Mode	Description
0	DISABLE	Output from the transmitter is passed unchanged to UARTn_TX
1	ENABLE	Output from the transmitter is inverted before it is passed to UARTn_TX

13	RXINV	0x0	RW	**Receiver Input Invert**

Setting this bit will invert the input to the EUART receiver.

Value	Mode	Description
0	DISABLE	Input is passed directly to the receiver
1	ENABLE	Input is inverted before it is passed to the receiver

12:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	MSBF	0x0	RW	**Most Significant Bit First**

Decides whether data is sent with the least significant bit first, or the most significant bit first.

Value	Mode	Description
0	DISABLE	Data is sent with the least significant bit first
1	ENABLE	Data is sent with the most significant bit first

9:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:5	OVS	0x0	RW	**Oversampling**

Sets the number of clock periods in a EUART bit-period. More clock cycles gives better robustness, while less clock cycles gives better performance.

Value	Mode	Description
0	X16	16X oversampling
1	X8	8X oversampling
2	X6	6X oversampling

Bit	Name	Reset	Access	Description
	3	X4		4X oversampling
	4	DISABLE		Disable oversampling (for LF operation)
4	MPAB	0x0	RW	**Multi-Processor Address-Bit**
	Defines the value of the multi-processor address bit. An incoming frame with its 9th bit equal to the value of this bit marks the frame as a multi-processor address frame.			
3	MPM	0x0	RW	**Multi-Processor Mode**
	Multi-processor mode uses the 9th bit of the EUART frames to tell whether the frame is an address frame or a data frame.			
	Value	Mode		Description
	0	DISABLE		The 9th bit of incoming frames has no special function
	1	ENABLE		An incoming frame with the 9th bit equal to MPAB will be loaded into the RX FIFO regardless of RXBLOCK and will result in the MPAB interrupt flag being set
2	CCEN	0x0	RW	**Collision Check Enable**
	Enables collision checking on data when operating in half duplex modus.			
	Value	Mode		Description
	0	DISABLE		Collision check is disabled
	1	ENABLE		Collision check is enabled. The receiver must be enabled for the check to be performed
1	LOOPBK	0x0	RW	**Loopback Enable**
	Allows the receiver to be connected directly to the EUART transmitter for loopback and half duplex communication.			
	Value	Mode		Description
	0	DISABLE		The receiver is connected to and receives data from UARTn_RX
	1	ENABLE		The receiver is connected to and receives data from UARTn_TX
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

22.5.4 EUSART_CFG1 - Configuration 1 Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0			0x0			0x0		0x0				0x0	0x0	0x0						0x0	0x0	0x0	0x0
Access									RW			RW			RW		RW				RW	RW	RW						RW	RW	RW	RW
Name									RTSRXFW			RXFIW			TXFIW		RXPRSEN				SFUBRX	RXDMAWU	TXDMAWU						RTSINV	CTSEN	CTSINV	DBGHALT

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:22	RTSRXFW	0x0	RW	**Request-to-send RX FIFO Watermark**

Set Request-to-send watermark level

Value	Mode	Description
0	ONEFRAME	RTS is set if there is space for at least one more frame in the RX FIFO.
1	TWOFRAMES	RTS is set if there is space for at least two more frames in the RX FIFO.
2	THREEFRAMES	RTS is set if there is space for at least three more frames in the RX FIFO.
3	FOURFRAMES	RTS is set if there is space for four more frames in the RX FIFO.

Bit	Name	Reset	Access	Description
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20:19	RXFIW	0x0	RW	**RX FIFO Interrupt Watermark**

Determines the interrupt and status level of the Receive FIFO. Also impacts RX DMA request.

Value	Mode	Description
0	ONEFRAME	RXFL status flag and IF are set when the RX FIFO has at least one frame in it.
1	TWOFRAMES	RXFL status flag and IF are set when the RX FIFO has at least two frames in it.
2	THREEFRAMES	RXFL status flag and IF are set when the RX FIFO has at least three frames in it.
3	FOURFRAMES	RXFL status flag and IF are set when the RX FIFO has four frames in it.

Bit	Name	Reset	Access	Description
18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	TXFIW	0x0	RW	**TX FIFO Interrupt Watermark**

Determines the interrupt and status level of the transmit FIFO. Also impacts TX DMA request.

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	ONEFRAME		TXFL status flag and IF are set when the TX FIFO has space for at least one more frame.
	1	TWOFRAMES		TXFL status flag and IF are set when the TX FIFO has space for at least two more frames.
	2	THREEFRAMES		TXFL status flag and IF are set when the TX FIFO has space for at least three more frames.
	3	FOURFRAMES		TXFL status flag and IF are set when the TX FIFO has space for at least four more frames.
15	RXPRSEN	0x0	RW	**PRS RX Enable**
	When set, the PRS channel selected as input to RX.			
14:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	SFUBRX	0x0	RW	**Start Frame Unblock Receiver**
	Set to unblock RX on Start frame reception.			
10	RXDMAWU	0x0	RW	**Receiver DMA Wakeup**
	Set to enable wakeup from EM2 to EM1 for DMA/ EUART RX interaction			
9	TXDMAWU	0x0	RW	**Transmitter DMA Wakeup**
	Set to enable wakeup from EM2 to EM1 for DMA/ EUART TX interaction			
8:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	RTSINV	0x0	RW	**Request-to-send Invert Enable**
	When set, the RTS pin polarity is inverted.			

Value	Mode	Description
0	DISABLE	The RTS pin is active low
1	ENABLE	The RTS pin is active high

2	CTSEN	0x0	RW	**Clear-to-send Enable**
	When set, frames in the TX FIFO will not be sent until link partner asserts CTS. Any data in the TX shift register will continue transmitting, the next TX FIFO data will not load into the TX shift register			

Value	Mode	Description
0	DISABLE	Ignore CTS
1	ENABLE	Stop transmitting when CTS is inactive

1	CTSINV	0x0	RW	**Clear-to-send Invert Enable**
	When set, the CTS pin polarity is inverted.			

Value	Mode	Description
0	DISABLE	The CTS pin is active low
1	ENABLE	The CTS pin is active high

0	DBGHALT	0x0	RW	**Debug halt**
	Check if sync type should be WSYNC			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	DISABLE	Continue normal UART operation even if core is halted
1	ENABLE	If core is halted, receive one frame and then halt reception by deactivating RTS. Next frame reception happens when the core is unhalted during single stepping.

22.5.5 EUSART_FRAMECFG - Frame Format Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x1				0x0								0x2	
Access																			RW				RW								RW	
Name																			STOPBITS				PARITY								DATABITS	

Bit	Name	Reset	Access	Description
31:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	STOPBITS	0x1	RW	**Stop-Bit Mode**

Determines the number of stop-bits used.

Value	Mode	Description
0	HALF	The transmitter generates a half stop bit. Stop-bits are not verified by receiver
1	ONE	One stop bit is generated and verified
2	ONEANDAHALF	The transmitter generates one and a half stop bit. The receiver verifies the first stop bit
3	TWO	The transmitter generates two stop bits. The receiver checks the first stop-bit only

11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	PARITY	0x0	RW	**Parity-Bit Mode**

Determines whether parity bits are enabled, and whether even or odd parity should be used.

Value	Mode	Description
0	NONE	Parity bits are not used
2	EVEN	Even parity are used. Parity bits are automatically generated and checked by hardware.
3	ODD	Odd parity is used. Parity bits are automatically generated and checked by hardware.

7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	DATABITS	0x2	RW	**Data-Bit Mode**

Sets the number of data bits in a EUART frame.

Value	Mode	Description
1	SEVEN	Each frame contains 7 data bits
2	EIGHT	Each frame contains 8 data bits
3	NINE	Each frame contains 9 data bits

Bit	Name	Reset	Access	Description

22.5.6 EUSART_IRHFCFG - HF IrDA Mod Config Register

Offset	Bit Position																															
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0	0x0		0x0
Access																													RW	RW		RW
Name																													IRHFFILT	IRHFPW		IRHFEN

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	IRHFFILT	0x0	RW	**IrDA RX Filter**
	Set to enable filter on demodulator.			

Value	Mode	Description
0	DISABLE	No filter enabled
1	ENABLE	Filter enabled. IrDA pulse must be high for at least 4 consecutive clock cycles to be detected

Bit	Name	Reset	Access	Description
2:1	IRHFPW	0x0	RW	**IrDA TX Pulse Width**
	Configure the pulse width generated by the modulator as a fraction of the configured EUART bit period.			

Value	Mode	Description
0	ONE	IrDA pulse width is 1/16 for OVS=0 and 1/8 for OVS=1
1	TWO	IrDA pulse width is 2/16 for OVS=0 and 2/8 for OVS=1
2	THREE	IrDA pulse width is 3/16 for OVS=0 and 3/8 for OVS=1
3	FOUR	IrDA pulse width is 4/16 for OVS=0 and 4/8 for OVS=1

Bit	Name	Reset	Access	Description
0	IRHFEN	0x0	RW	**Enable IrDA Module**
	Enable IrDA module and route EUART signals through it. Used when EUART has HF clock.			

22.5.7 EUSART_IRLFCFG - LF IrDA Pulse Config Register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																IRLFEN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	IRLFEN	0x0	RW	**Pulse Generator/Extender Enable**
	Filter EUART output through pulse generator and the EUART input through the pulse extender. Used for LF operation.			

22.5.8 EUSART_TIMINGCFG - Timing Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	
Access																															RW	
Name																															TXDELAY	

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	TXDELAY	0x0	RW	**TX Delay Transmission**
	Configurable delay before new rtansfers. Frames sent back-to-back are not delayed.			

Value	Mode	Description
0	NONE	Frames are transmitted immediately.
1	SINGLE	Transmission of new frames is delayed by a single bit period.
2	DOUBLE	Transmission of new frames is delayed by a two bit periods.
3	TRIPPLE	Transmission of new frames is delayed by a three bit periods.

22.5.9 EUSART_STARTFRAMECFG - Start Frame Register

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												RW				
Name																												STARTFRAME				

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	STARTFRAME	0x0	RW	**Start Frame**
	When a frame matching STARTFRAME is received, the receiver detects that and STARTF interrupt flag is set. If SFUBRX is set, RXBLOCK is cleared and the start frame is loaded in to the RX FIFO.			

22.5.10 EUSART_SIGFRAMECFG - Signal Frame Register

Offset	Bit Position																															
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												RW				
Name																												SIGFRAME				

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	SIGFRAME	0x0	RW	**Signal Frame Value**
	When a frame matching SIGFRAME is detected by the receiver, SIGF interrupt flag is set.			

22.5.11 EUSART_CLKDIV - Clock Divider Register

Offset	Bit Position																															
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset										0x0																						
Access										RW																						
Name										DIV																						

Bit	Name	Reset	Access	Description
31:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:3	DIV	0x0	RW	**Fractional Clock Divider**
	Specifies the fractional clock divider for EUART. Bits [7:3] are the fractional part and bits [22:8] are the integer part. The total divider is ([22:8] + [7:3]/32). To make the math easier the total divider can also be calculated as ([22:8] + [7:0]/256) where bits [0:2] will always be 0. Setting AUTOBAUDEN in CFG0 register will overwrite the DIV field.			
2:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

22.5.12 EUSART_TRIGCTRL - Trigger Control Register

Offset	Bit Position																															
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															TXTEN	RXTEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	TXTEN	0x0	RW	**Transmit Trigger Enable**
	When set, the PRS channel selected by TSEL sets TXEN, enabling the transmitter on positive trigger edges.			
0	RXTEN	0x0	RW	**Receive Trigger Enable**
	When set, the PRS channel selected by TSEL sets RXEN, enabling the receiver on positive trigger edges.			

22.5.13 EUSART_CMD - Command Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																								W	W	W	W	W	W	W	W	W
Name																								CLEARTX	TXTRIDIS	TXTRIEN	RXBLOCKDIS	RXBLOCKEN	TXDIS	TXEN	RXDIS	RXEN

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	CLEARTX	0x0	W	**Clear TX FIFO**
	Set to clear TX FIFO. EUART must be enabled (EUARTn_EN=1) but should not be transmitting (can be achieved by disabling transmitter).			
7	TXTRIDIS	0x0	W	**Transmitter Tristate Disable**
	Disables tristating of the transmitter output.			
6	TXTRIEN	0x0	W	**Transmitter Tristate Enable**
	Tristates the transmitter output.			
5	RXBLOCKDIS	0x0	W	**Receiver Block Disable**
	Set to clear RXBLOCK, resulting in all incoming frames being loaded into the RX FIFO			
4	RXBLOCKEN	0x0	W	**Receiver Block Enable**
	Set to set RXBLOCK, resulting in all incoming frames being discarded.			
3	TXDIS	0x0	W	**Transmitter Disable**
	Set to disable transmission. Any on going transmission will be aborted.			
2	TXEN	0x0	W	**Transmitter Enable**
	Set to enable data transmission.			
1	RXDIS	0x0	W	**Receiver Disable**
	Set to disable data reception. If a frame is under reception when the receiver is disabled, the incoming frame is discarded.			
0	RXEN	0x0	W	**Receiver Enable**
	Set to activate data reception on EUARTn_RX.			

22.5.14 EUSART_RXDATA - RX Data Register

Offset	Bit Position																															
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x0	0x0	0x0								
Access																						R	R	R								
Name																						FERR	PERR	RXDATA								

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	FERR	0x0	R	**Framing Error**
	Set if received data has a framing error. Can be the result of a break condition.			
9	PERR	0x0	R	**Parity Error**
	Set if received data has a parity error.			
8:0	RXDATA	0x0	R	**RX Data**
	Use this register to read data from the EUART buffer.			

22.5.15 EUSART_RXDATAP - RX Data Peek Register

Offset	Bit Position																															
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x0	0x0	0x0								
Access																						R	R	R								
Name																						FERRP	PERRP	RXDATAP								

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	FERRP	0x0	R	**Framing Error Peek**
	Set if received data has a framing error. Can be the result of a break condition.			
9	PERRP	0x0	R	**Parity Error Peek**
	Set if received data has a parity error.			
8:0	RXDATAP	0x0	R	**RX Data Peek**
	Use this register to access data read from the EUART without popping the FIFO.			

22.5.16 EUSART_TXDATA - TX Data Register

Offset	Bit Position																													
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x0	0x0	0x0	0x0	0x0	0x0								
Access																			W	W	W	W	W	W								
Name																			RXENAT	TXDISAT	TXBREAK	TXTRIAT	UBRXAT	TXDATA								

Bit	Name	Reset	Access	Description
31:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	RXENAT	0x0	W	**Enable RXEN After Transmission**
	Set to enable reception after transmission.			
12	TXDISAT	0x0	W	**Clear TXEN After Transmission**
	Set to disable trasmitter and release data bus directly after transmission.			
11	TXBREAK	0x0	W	**Transit Data as Break**
	Set to send data as a break. Recipient will see a framing error or a break condition depending on its configuration and the value of TXDATA.			
10	TXTRIAT	0x0	W	**Set TXTRI After Transmisssion**
	Set to tri-state transmitter by setting TXTRI after transmission.			
9	UBRXAT	0x0	W	**Unblock RX After Transmission**
	Set to clear RXBLOCK after transmission, unblocking the receiver.			
8:0	TXDATA	0x0	W	**TX Data**
	Use this register to write data to the EUART. If TXEN is set, a transfer will be initiated at the first opportunity.			

22.5.17 EUSART_STATUS - Status Register

Offset	Bit Position																													
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x0					0x0		0x0				0x1	0x1				0x0	0x0	0x1	0x0	0x0	0x0		0x0	0x0
Access								R					R		R				R	R				R	R	R	R	R	R		R	R
Name								AUTOBAUDDONE					CLEARTXBUSY		TXFCNT				TXIDLE	RXIDLE				RXFULL	RXFL	TXFL	TXC	TXTRI	RXBLOCK		TXENS	RXENS

Bit	Name	Reset	Access	Description
31:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	AUTOBAUDDONE	0x0	R	**Auto Baud Rate Detection Completed**
	Set when auto baud rate has been detected and CLKDIV has been updated with required value. If AUTOBAUDEN is not set in CFG0 register, this bit is always read as '0'.			
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	CLEARTXBUSY	0x0	R	**TX FIFO Clear Busy**
	After issuing CLEARTX command, wait on this status flag until it goes low			
18:16	TXFCNT	0x0	R	**Valid entries in TX FIFO**
	Count of TX valid FIFO entries			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	TXIDLE	0x1	R	**TX Idle**
	Set when TX idle			
12	RXIDLE	0x1	R	**RX Idle**
	Set when RX is idle			
11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	RXFULL	0x0	R	**RX FIFO Full**
	Set when the RX FIFO is full.			
7	RXFL	0x0	R	**RX FIFO Level**
	Set when data is available in the RX FIFO. Depends on the RXFIW setting in the CFG1 register.			
6	TXFL	0x1	R	**TX FIFO Level**
	Set when there is space for data in the TX FIFO. Depends on the TXFIW setting in CFG1 register.			
5	TXC	0x0	R	**TX Complete**
	Set when a transmission has completed and no more data is available in the TX FIFO and shift register.			
4	TXTRI	0x0	R	**Transmitter Tristated**

Bit	Name	Reset	Access	Description
	Set when the transmitter is tristated, and cleared when transmitter output is enabled. If AUTOTRI in UARTn_CFG is set, then this bit is always read as 0.			
3	RXBLOCK	0x0	R	**Block Incoming Data**
	When set, the receiver discards incoming frames. An incoming frame will not be loaded into the RX FIFO if this bit is set at the instant the frame has been completely received.			
2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	TXENS	0x0	R	**Transmitter Enable Status**
	Set when the transmitter is enabled.			
0	RXENS	0x0	R	**Receiver Enable Status**
	Set when the receiver is enabled.			

22.5.18 EUSART_IF - Interrupt Flag Register

Offset	Bit Position																													
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x0					0x0	0x0					0x0	0x0		0x0	0x0	0x0		0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access								RW					RW	RW					RW	RW		RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
Name								AUTOBAUDDONE					SIGF	STARTF					TXIDLE	CCF		MPAF	FERR	PERR		TXOF	RXUF	RXOF	RXFULL	RXFL	TXFL	TXC

Bit	Name	Reset	Access	Description
31:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	AUTOBAUDDONE	0x0	RW	**Auto Baud Complete Interrupt Flag**
	Set when auto baud rate detection is complete.			
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	SIGF	0x0	RW	**Signal Frame Interrupt Flag**
	Set when a signal frame is detected. Please note that when MPA, START, and SIGNAL are set to match the same frame, corresponding interrupts might get triggered in arbitrary sequence due to synchronization uncertainty.			
18	STARTF	0x0	RW	**Start Frame Interrupt Flag**
	Set when a start frame is detected. Please note that when MPA, START, and SIGNAL are set to match the same frame, corresponding interrupts might get triggered in arbitrary sequence due to synchronization uncertainty.			
17:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	TXIDLE	0x0	RW	**TX Idle Interrupt Flag**
	Set when TX goes idle. At this point, transmission has ended			
12	CCF	0x0	RW	**Collision Check Fail Interrupt Flag**
	Set when a collision check notices an error in the transmitted data.			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	MPAF	0x0	RW	**Multi-Processor Address Frame Interrupt**
	Set when a multi-processor address frame is detected.			
9	FERR	0x0	RW	**Framing Error Interrupt Flag**
	Set when a frame with a framing error is received while RXBLOCK is cleared.			
8	PERR	0x0	RW	**Parity Error Interrupt Flag**
	Set when a frame with a parity error is received while RXBLOCK is cleared.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	TXOF	0x0	RW	**TX FIFO Overflow Interrupt Flag**

Bit	Name	Reset	Access	Description
	Set when a write is done to the TX FIFO while it is full. The data already in the TX FIFO is preserved. Also set if DMA tries to write to the TX FIFO while a CLEARTX command is running.			
5	RXUF	0x0	RW	**RX FIFO Underflow Interrupt Flag**
	Set when trying to read from the RX FIFO when it is empty.			
4	RXOF	0x0	RW	**RX FIFO Overflow Interrupt Flag**
	Set when data is complettly received in the receive shift register but the RX FIFO is full. RX FIFO is not overwirtten by new data.			
3	RXFULL	0x0	RW	**RX FIFO Full Interrupt Flag**
	Set when the RX FIFO becomes full.			
2	RXFL	0x0	RW	**RX FIFO Level Interrupt Flag**
	Set when data becomes available in the RX FIFO. This field depends on the RXFIW field in the CFG1 register.			
1	TXFL	0x0	RW	**TX FIFO Level Interrupt Flag**
	Set when space becomes available in the TX FIFO. This depends on the TXFIW field in the CFG1 register.			
0	TXC	0x0	RW	**TX Complete Interrupt Flag**
	This interrupt is set after a transmission when both the TX FIFO and shift register are empty.			

22.5.19 EUSART_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x048	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x0					0x0	0x0					0x0	0x0		0x0	0x0	0x0		0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access								RW					RW	RW					RW	RW		RW	RW	RW		RW	RW	RW	RW	RW	RW	RW
Name								AUTOBAUDDONE					SIGF	STARTF					TXIDLE	CCF		MPAF	FERR	PERR		TXOF	RXUF	RXOF	RXFULL	RXFL	TXFL	TXC

Bit	Name	Reset	Access	Description
31:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	AUTOBAUDDONE	0x0	RW	**Auto Baud Complete IEN**
	Interrupt enable for AUTOBAUDDONEIF.			
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	SIGF	0x0	RW	**Signal Frame IEN**
	Interrupt enable for SIGFIF.			
18	STARTF	0x0	RW	**Start Frame IEN**
	Interrupt enable for STARTFIF.			
17:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	TXIDLE	0x0	RW	**TX IDLE IEN**
	Interrupt enable for TXIDLEIF.			
12	CCF	0x0	RW	**Collision Check Fail IEN**
	Interrupt enable for CCFIF.			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	MPAF	0x0	RW	**Multi-Processor Addr Frame IEN**
	Interrupt enable for MPAFIF.			
9	FERR	0x0	RW	**Framing Error IEN**
	Interrupt enable for FERRIF.			
8	PERR	0x0	RW	**Parity Error IEN**
	Interrupt enable for PERRIF.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	TXOF	0x0	RW	**TX FIFO Overflow IEN**
	Interrupt enable for TXOFIF.			

Bit	Name	Reset	Access	Description
5	RXUF	0x0	RW	**RX FIFO Underflow IEN**
	Interrupt enable for RXUFIF.			
4	RXOF	0x0	RW	**RX FIFO Overflow IEN**
	Interrupt enable for RXOFIF.			
3	RXFULL	0x0	RW	**RX FIFO Full IEN**
	Interrupt enable for RXFULLIF.			
2	RXFL	0x0	RW	**RX FIFO Level IEN**
	Interrupt enable for RXFLIF.			
1	TXFL	0x0	RW	**TX FIFO Level IEN**
	Interrupt enable for TXFLIF.			
0	TXC	0x0	RW	**TX Complete IEN**
	Interrupt enable for TXCIF.			

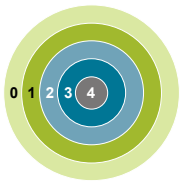
22.5.20 EUSART_SYNCBUSY - Synchronization Busy Register

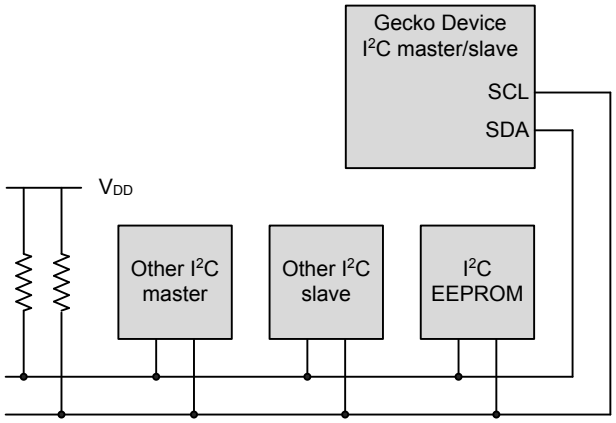
Offset	Bit Position																												
0x04C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																						R	R	R	R	R	R	R	R	R	R	R
Name																						TXTRIDIS	TXTRIEN	RXBLOCKDIS	RXBLOCKEN	TXDIS	TXEN	RXDIS	RXEN	TXTEN	RXTEN	DIV

Bit	Name	Reset	Access	Description
31:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	TXTRIDIS	0x0	R	**SYNCBUSY in TXTRIDIS in CMD**
	This bit is set when there is an ongoing synchronization of TXTRIDIS field. Do not do another write to the same field while this bit is set.			
9	TXTRIEN	0x0	R	**SYNCBUSY for TXTRIEN in CMD**
	This bit is set when there is an ongoing synchronization of TXTRIEN field. Do not do another write to the same field while this bit is set.			
8	RXBLOCKDIS	0x0	R	**SYNCBUSY for RXBLOCKDIS in CMD**
	This bit is set when there is an ongoing synchronization of RXBLOCKDIS field. Do not do another write to the same field while this bit is set.			
7	RXBLOCKEN	0x0	R	**SYNCBUSY for RXBLOCKEN in CMD**
	This bit is set when there is an ongoing synchronization of RXBLOCKEN field. Do not do another write to the same field while this bit is set.			
6	TXDIS	0x0	R	**SYNCBUSY for TXDIS in CMD**
	This bit is set when there is an ongoing synchronization of TXDIS field. Do not do another write to the same field while this bit is set.			
5	TXEN	0x0	R	**SYNCBUSY for TXEN in CMD**
	This bit is set when there is an ongoing synchronization of TXEN field. Do not do another write to the same field while this bit is set.			
4	RXDIS	0x0	R	**SYNCBUSY for RXDIS in CMD**
	This bit is set when there is an ongoing synchronization of RXDIS field. Do not do another write to the same field while this bit is set.			
3	RXEN	0x0	R	**SYNCBUSY for RXEN in CMD**
	This bit is set when there is an ongoing synchronization of RXEN field. Do not do another write to the same field while this bit is set.			
2	TXTEN	0x0	R	**SYNCBUSY for TXTEN in TRIGCTRL**
	This bit is set when there is an ongoing synchronization of TXTEN field. Do not do another write to the same field while this bit is set.			
1	RXTEN	0x0	R	**SYNCBUSY for RXTEN in TRIGCTRL**
	This bit is set when there is an ongoing synchronization of RXTEN field. Do not do another write to the same field while this bit is set.			

Bit	Name	Reset	Access	Description
0	DIV	0x0	R	**SYNCBUSY for DIV in CLKDIV** This bit is set when there is an ongoing synchronization of DIV field. Do not do another write to the same field while this bit is set.

23. I2C - Inter-Integrated Circuit Interface





Quick Facts

What?

The I²C interface allows communication on I²C-buses with the lowest energy consumption possible.

Why?

I²C is a popular serial bus that enables communication with a number of external devices using only two I/O pins.

How?

With the help of DMA, the I²C interface allows I²C communication with minimal CPU intervention. Address recognition is available in all energy modes (except EM4), allowing the MCU to wait for data on the I²C-bus with sub-µA current consumption.

23.1 Introduction

The I²C module provides an interface between the MCU and a serial I²C-bus. It is capable of acting as both a master and a slave and supports multi-master buses. Standard-mode, fast-mode and fast-mode plus speeds are supported, allowing transmission rates all the way from 10 kbit/s up to 1 Mbit/s. Slave arbitration and timeouts are also provided to allow implementation of an SMBus compliant system. The interface provided to software by the I²C module allows precise control of the transmission process and highly automated transfers. Automatic recognition of slave addresses is provided in all energy modes (except EM4).

23.2 Features

- True multi-master capability
- Support for different bus speeds
  - Standard-mode (Sm) bit rate up to 100 kbit/s
  - Fast-mode (Fm) bit rate up to 400 kbit/s
  - Fast-mode Plus (Fm+) bit rate up to 1 Mbit/s
- Arbitration for both master and slave (allows SMBus ARP)
- Clock synchronization and clock stretching
- Hardware address recognition
  - 7-bit masked address
  - General call address
  - Supported in EM2/3 (I2C0-only)
- 10-bit address support
- Error handling
  - Clock low timeout
  - Clock high timeout
  - Arbitration lost
  - Bus error detection
- Separate receive/ transmit 2-level buffers, with additional separate shift registers
- Full DMA support

23.3 Functional Description

An overview of the I2C module is shown in Figure 23.1 I2C Overview on page 676.

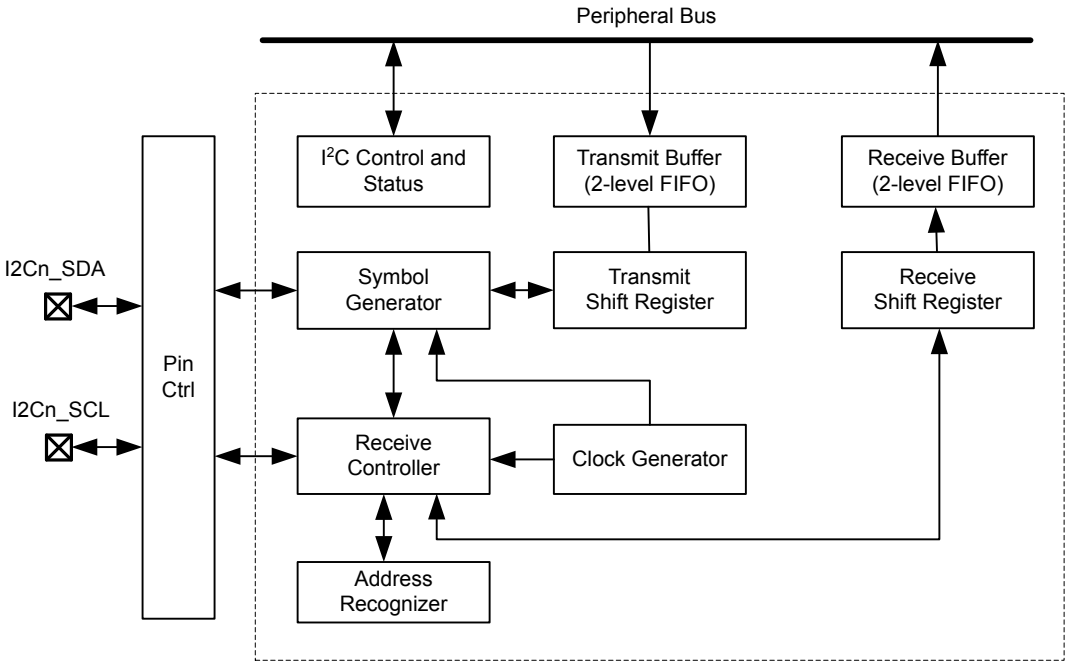


Figure 23.1. I2C Overview

23.3.1 I2C-Bus Overview

The I²C-bus uses two wires for communication; a serial data line (SDA) and a serial clock line (SCL) as shown in Figure 23.2 I2C-Bus Example on page 677. As a true multi-master bus it includes collision detection and arbitration to resolve situations where multiple masters transmit data at the same time without data loss.

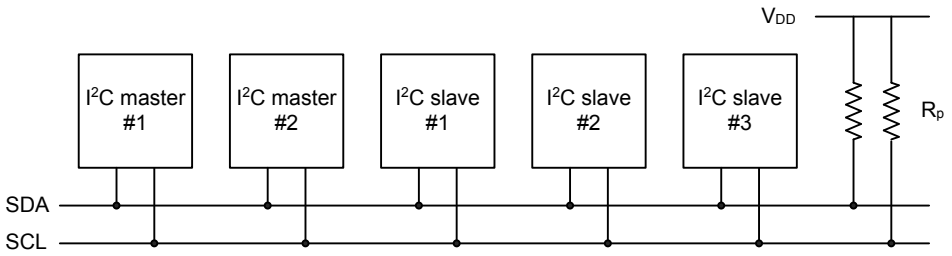


Figure 23.2. I2C-Bus Example

Each device on the bus is addressable by a unique address, and an I²C master can address all the devices on the bus, including other masters.

Both the bus lines are open-drain. The maximum value of the pull-up resistor can be calculated as a function of the maximal rise-time **tr** for the given bus speed, and the estimated bus capacitance **Cb** as shown in Figure 23.3 I2C Pull-up Resistor Equation on page 677.

$$R_{p(max)} = t_r/(0.8473 \times C_b).$$

Figure 23.3. I2C Pull-up Resistor Equation

The maximal rise times for 100 kHz, 400 kHz and 1 MHz I²C are 1 µs, 300 ns and 120 ns respectively.

Note: The GPIO slew rate control should be set for the desired slew rate..

Note: If V_{dd} drops below the voltage on SCL and SDA lines, the MCU could become back powered and pull the SCL and SDA lines low.

23.3.1.1 START and STOP Conditions

START and STOP conditions are used to initiate and stop transactions on the I²C-bus. All transactions on the bus begin with a START condition (S) and end with a STOP condition (P). As shown in Figure 23.4 I2C START and STOP Conditions on page 678, a START condition is generated by pulling the SDA line low while SCL is high, and a STOP condition is generated by pulling the SDA line high while SCL is high.

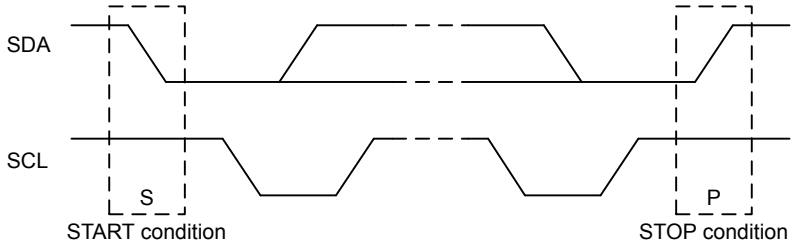


Figure 23.4. I2C START and STOP Conditions

The START and STOP conditions are easily identifiable bus events as they are the only conditions on the bus where a transition is allowed on SDA while SCL is high. During the actual data transmission, SDA is only allowed to change while SCL is low, and must be stable while SCL is high. One bit is transferred per clock pulse on the I²C-bus as shown in Figure 23.5 I2C Bit Transfer on I²C-Bus on page 678.

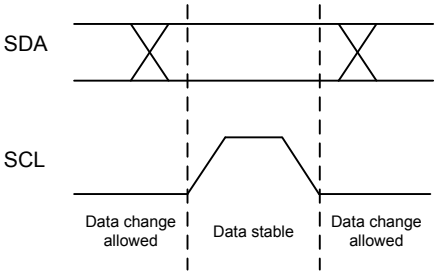


Figure 23.5. I2C Bit Transfer on I²C-Bus

23.3.1.2 Bus Transfer

When a master wants to initiate a transfer on the bus, it waits until the bus is idle and transmits a START condition on the bus. The master then transmits the address of the slave it wishes to interact with and a single R/W bit telling whether it wishes to read from the slave (R/W bit set to 1) or write to the slave (R/W bit set to 0).

After the 7-bit address and the R/W bit, the master releases the bus, allowing the slave to acknowledge the request. During the next bit-period, the slave pulls SDA low (ACK) if it acknowledges the request, or keeps it high if it does not acknowledge it (NACK).

Following the address acknowledge, either the slave or master transmits data, depending on the value of the R/W bit. After every 8 bits (one byte) transmitted on the SDA line, the transmitter releases the line to allow the receiver to transmit an ACK or a NACK. Both the data and the address are transmitted with the most significant bit first.

The number of bytes in a bus transfer is unrestricted. The master ends the transmission after a (N)ACK by sending a STOP condition on the bus. After a STOP condition, any master wishing to initiate a transfer on the bus can try to gain control of it. If the current master wishes to make another transfer immediately after the current, it can start a new transfer directly by transmitting a repeated START condition (Sr) instead of a STOP followed by a START.

Examples of I^2C transfers are shown in Figure 23.6 I2C Single Byte Write to Slave on page 679, Figure 23.7 I2C Double Byte Read from Slave on page 679, and Figure 23.8 I2C Single Byte Write, then Repeated Start and Single Byte Read on page 679. The identifiers used are:

- ADDR - Address
- DATA - Data
- S - Start bit
- Sr - Repeated start bit
- P - Stop bit
- W/R - Read(1)/Write(0)
- A - ACK
- N - NACK

Figure 23.6. I2C Single Byte Write to Slave

Figure 23.7. I2C Double Byte Read from Slave

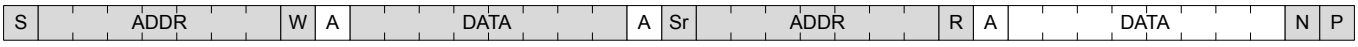


Figure 23.8. I2C Single Byte Write, then Repeated Start and Single Byte Read

23.3.1.3 Addresses

I²C supports both 7-bit and 10-bit addresses. When using 7-bit addresses, the first byte transmitted after the START-condition contains the address of the slave that the master wants to contact. In the 7-bit address space, several addresses are reserved. These addresses are summarized in Table 23.1 I2C Reserved I²C Addresses on page 680, and include a General Call address which can be used to broadcast a message to all slaves on the I²C-bus.

Table 23.1. I2C Reserved I²C Addresses

I²C Address	R/W	Description
0000-000	0	General Call address
0000-000	1	START byte
0000-001	X	Reserved for the C-Bus format
0000-010	X	Reserved for a different bus format
0000-011	X	Reserved for future purposes
0000-1XX	X	Reserved for future purposes
1111-1XX	X	Reserved for future purposes
1111-0XX	X	10 Bit slave addressing mode

23.3.1.4 10-bit Addressing

To address a slave using a 10-bit address, two bytes are required to specify the address instead of one. The seven first bits of the first byte must then be 1111 0XX, where XX are the two most significant bits of the 10-bit address. As with 7-bit addresses, the eighth bit of the first byte determines whether the master wishes to read from or write to the slave. The second byte contains the eight least significant bits of the slave address.

When a slave receives a 10-bit address, it must acknowledge both the address bytes if they match the address of the slave.

When performing a master transmitter operation, the master transmits the two address bytes and then the remaining data, as shown in Figure 23.9 I2C Master Transmitter/Slave Receiver with 10-bit Address on page 680.

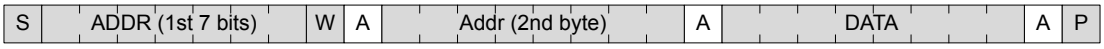


Figure 23.9. I2C Master Transmitter/Slave Receiver with 10-bit Address

When performing a master receiver operation however, the master first transmits the two address bytes in a master transmitter operation, then sends a repeated START followed by the first address byte and then receives data from the addressed slave. The slave addressed by the 10-bit address in the first two address bytes must remember that it was addressed, and respond with data if the address transmitted after the repeated start matches its own address. An example of this (with one byte transmitted) is shown in Figure 23.10 I2C Master Receiver/Slave Transmitter with 10-bit Address on page 680.

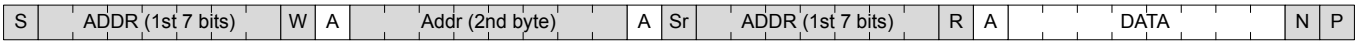


Figure 23.10. I2C Master Receiver/Slave Transmitter with 10-bit Address

23.3.1.5 Arbitration, Clock Synchronization, Clock Stretching

Arbitration and clock synchronization are features aimed at allowing multi-master buses. Arbitration occurs when two devices try to drive the bus at the same time. If one device drives it low, while the other drives it high, the one attempting to drive it high will not be able to do so due to the open-drain bus configuration. Both devices sample the bus, and the one that was unable to drive the bus in the desired direction detects the collision and backs off, letting the other device continue communication on the bus undisturbed.

Clock synchronization is a means of synchronizing the clock outputs from several masters driving the bus at once, and is a requirement for effective arbitration.

Slaves on the bus are allowed to force the clock output on the bus low in order to pause the communication on the bus and give themselves time to process data or perform any real-time tasks they might have. This is called clock stretching.

Arbitration is supported by the I^{2}C module for both masters and slaves. Clock synchronization and clock stretching is also supported.

23.3.2 Enable and Reset

The I^{2}C is enabled by setting the EN bit in the I2C_EN register.

To reset the internal state of the I^{2}C module and terminate any ongoing transfers, set the CORERST bit in I2C_CTRL. After resetting, the CORERST bit must be cleared to resume I^{2}C operation.

Note: When enabling the I^{2}C, the ABORT command or the Bus Idle Timeout feature must be applied prior to use even if the BUSY flag is not set.

23.3.3 Pin Configuration

The I^{2}C SDA and SCL pins are configured and enableed in the GPIO_DBUSI2Cn_ROUTEEN, GPIO_DBUSI2Cn_SCLROUTE, and GPIO_DBUSI2Cn_SDAROUTE registers.

The I^{2}C module must be configured to use pins on either Port A or B if wakeup on address recognition from EM2/3 is desired. All other ports are available only in EM0/1. See GPIO chapter for more details on Port limitations.

If the I^{2}C module is configured to use pins other than Port A or B, firmware should reset the module before entering EM2/3 by setting the CORERST bit in I2C_CTRL. After resuming EM0/1 operation, firmware should then clear CORERST.

23.3.4 Safely Disabling and Changing Slave Configuration

The I^{2}C slave is partially asynchronous, and some precautions are necessary to always ensure a safe slave disable or slave configuration change. These measures should be taken, if (while the slave is enabled) the user cannot guarantee that an address match will not occur at the exact time of slave disable or slave configuration change.

Worst case consequences for an address match while disabling slave or changing configuration is that the slave may end up in an undefined state. To reset the slave back to a known state, the EN bit in I2C_EN must be cleared. This should be done regardless of whether the slave is going to be re-enabled or not.

23.3.5 Clock Generation

The I^2C peripheral clock (I2CCLK) for I2C0 is derived from the LSPCLK (max freq = 25 MHz), and for I2C1 is derived from the PCLK (max freq = 50 MHz).

The SCL signal generated by the I^2C master determines the maximum transmission rate on the bus. The clock is generated as a division of the peripheral clock (I2CCLK), and is given by the following equation:

$$f_{SCL} = f_{I2CCLK}/(((N_{low} + N_{high}) \times (DIV + 1)) + 8),$$

Figure 23.11. I2C Maximum Transmission Rate

Where DIV is the clock divider value set in I2C_CLKDIV, and the values of N_{low} and N_{high} (and thus the ratio between the high and low parts of the clock signal) are controlled by CLHR in the I2C_CTRL register.

The values of N_{low} and N_{high}, in combination with the synchronization cycles (discussed below), specify the number of prescaled clock cycles in the low and high periods of the clock signal respectively. The worst case low and high periods of the signal are:

$$T_{high} >= ((N_{high}) \times (DIV + 1) + 4)/f_{I2CCLK},$$

$$T_{low} >= (N_{low} \times (DIV + 1) + 4)/f_{I2CCLK}.$$

Figure 23.12. I2C High and Low Cycles Equations

In worst case, T_{high} and T_{low} can be 1 f_{I2CCLK} cycle longer than the number found by above equations due to synchronization uncertainty (i.e., if the synchronization takes 3 f_{I2CCLK} cycles instead of 2). Similarly, in the worst case the number 8 in the denominator in f_{SCL} equation can be 9 (if the synchronization cycles were 3 instead of 2 in T_{high} or T_{low}) or 10 (if synchronization cycles were 3 in both T_{high} and T_{low}).

Note: DIV must be set to 1 during slave mode operation.

23.3.6 Arbitration

Arbitration is enabled by default, but can be disabled by setting the ARBDIS bit in I2C_CTRL. When arbitration is enabled, the value on SDA is sensed every time the I^2C module attempts to change its value. If the sensed value is different than the value the I^2C module tried to output, it is interpreted as a simultaneous transmission by another device, and that the I^2C module has lost arbitration.

Whenever arbitration is lost, the ARBLOST interrupt flag in I2C_IF is set, any lines held are released, and the I^2C device goes idle. If an I^2C master loses arbitration during the transmission of an address, another master may be trying to address it. The master therefore receives the rest of the address, and if the address matches the slave address of the master, the master goes into either slave transmitter or slave receiver mode.

Note:

Arbitration can be lost both when operating as a master and when operating as a slave.

23.3.7 Buffers

23.3.7.1 Transmit Buffer and Shift Register

The I²C transmitter has a 2-level FIFO transmit buffer and a transmit shift register as shown in Figure 23.1 I2C Overview on page 676. A byte is loaded into the transmit buffer by writing to I2C_TXDATA or 2 bytes can be loaded simultaneously in the transmit buffer by writing to I2C_TXDOUBLE. Figure 23.13 I2C Transmit Buffer Operation on page 683 shows the basics of the transmit buffer. When the transmit shift register is empty and ready for new data, the byte from the transmit buffer is then loaded into the shift register. The byte is then kept in the shift register until it is transmitted. When a byte has been transmitted, a new byte is loaded into the shift register (if available in the transmit buffer). If the transmit buffer is empty, then the shift register also remains empty. The TXC flag in I2C_STATUS and the TXC interrupt flags in I2C_IF are then set, signaling that the transmit shift register is out of data. TXC is cleared when new data becomes available, but the TXC interrupt flag must be cleared by software.

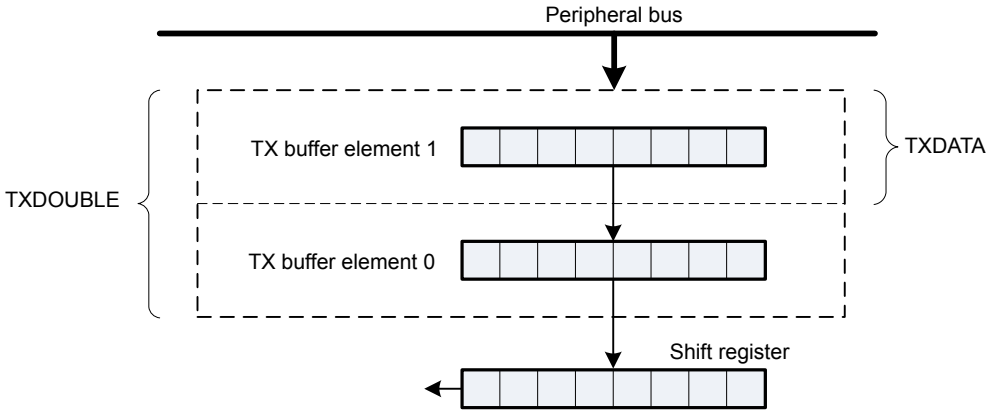


Figure 23.13. I2C Transmit Buffer Operation

The TXBL flags in I2C_STATUS and I2C_IF are used to indicate the level of the transmit buffer. The TXBIL bit in I2C_CTRL controls the level at which these flag bits are set:

- If TXBIL is cleared, the TXBL flags are set whenever the transmit buffer becomes empty (used when transmitting using I2C_TXDOUBLE).
- If TXBIL is set, the TXBL flags are set whenever the transmit buffer goes from full to half-empty or empty (used when transmitting with I2C_TXDATA).

The TXBL status flag in I2C_STATUS is cleared automatically when the condition becomes false. After the transmit FIFOs are filled, software needs to manually clear the TXBL interrupt flag. Note that the TXBL interrupt flag is 0 by default, but immediately after software sets I2C_EN.EN = 1, the TXBL interrupt flag will be set to indicate the transmit FIFO is empty. When the I²C module is disabled (I2C_EN.EN= 0), software needs to manually clear the TXBL interrupt flag (or ignore it).

Additionally, the TXBUFCNT bitfield in I2C_STATUS can be read to determine the exact number of transmit buffers filled with valid data. This is particularly useful for determining whether the transmit buffers are full. For example, if TXBUFCNT = '2', firmware can determine that both transmit buffers are filled, and that any additional data written to the transmit buffer would result in an overflow condition. Note that the TXBUFCNT count does not include the TX shift register.

If an attempt is made to write more bytes to the transmit buffer than the space available, the TXOF interrupt flag in I2C_IF is set, indicating the overflow. The data already in the buffer remains preserved, and no new data is written.

The transmit buffer and the transmit shift register can be cleared by setting command bit CLEARTX in I2C_CMD. This will prevent the I²C module from transmitting the data in the buffer and the shift register, and will make them available for new data. Any byte currently being transmitted will not be aborted. Transmission of this byte will be completed.

23.3.7.2 Receive Buffer and Shift Register

The I^{2}C receiver uses a 2-level FIFO receive buffer and a receive shift register as shown in Figure 23.14 I2C Receive Buffer Operation on page 684. When a byte has been fully received by the receive shift register, it is loaded into the receive buffer if there is room for it, making the shift register empty to receive another byte. Otherwise, the byte waits in the shift register until space becomes available in the buffer.

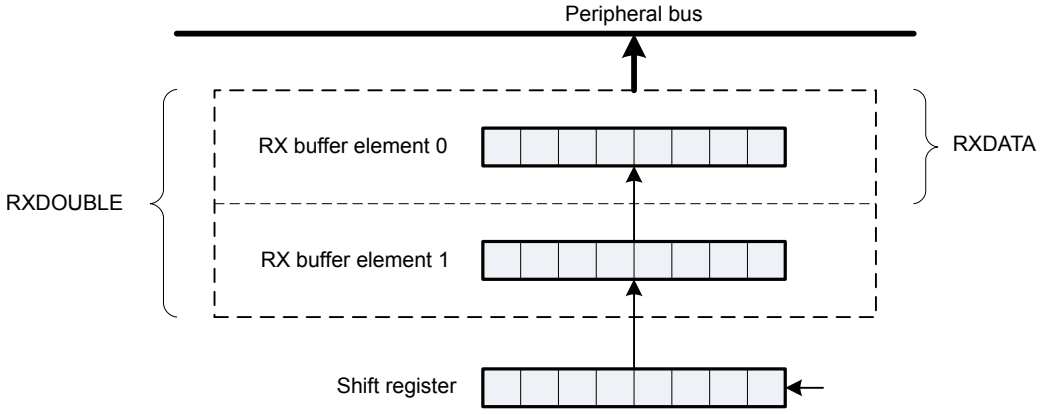


Figure 23.14. I2C Receive Buffer Operation

When a byte becomes available in the receive buffer, the RXDATAV flags in I2C_STATUS and I2C_IF are set. When the buffer becomes full, the RXFULL flags in I2C_STATUS and I2C_IF are set. The RXDATAV and RXFULL flags in I2C_STATUS are automatically cleared by hardware when their condition is no longer true. The RXDATAV and RXFULL flags in I2C_IF must be manually cleared by software after the receive FIFO is emptied. Note that when the RXFULL flag is set, indicating the buffer is full, space is still available in the receive shift register for one more byte.

The data can be fetched from the buffer in two ways. I2C_RXDATA gives access to the received byte (if two bytes are received then the one received first is fetched first). I2C_RXDOUBLE makes it possible to read the two received bytes simultaneously. If an attempt is made to read more bytes from the buffer than available, the RXUF interrupt flag in I2C_IF is set to signal the underflow, and the data read from the buffer is undefined.

When using I2C_RXDOUBLE to pick data, AUTOACK in I2C_CTRL should be set to 1. This ensures that an ACK is automatically sent out after the first byte is received so that the reception of the next byte can begin. In order to stop receiving data bytes, a NACK must be sent out through the I2C_CMD register.

I2C_RXDATAP and I2C_RXDOUBLEP can be used to read data from the receive buffer without removing it from the buffer. The RXUF interrupt flag in I2C_IF will never be set as a result of reading from I2C_RXDATAP and I2C_RXDOUBLEP, but the data read through I2C_RXDATAP when the receive buffer is empty is still undefined.

Once a transaction is complete (STOP sent or received), the receive buffer needs to be flushed (all received data must be read) before starting a new transaction.

23.3.8 Master Operation

A bus transaction is initiated by transmitting a START condition (S) on the bus. This is done by setting the START bit in I2C_CMD. The command schedules a START condition, and makes the I^2C module generate a start condition whenever the bus becomes free.

The I^2C-bus is considered busy whenever another device on the bus transmits a START condition. Until a STOP condition is detected, the bus is owned by the master issuing the START condition. The bus is considered free when a STOP condition is transmitted on the bus. After a STOP is detected, all masters that have data to transmit send a START condition and begin transmitting data. Arbitration ensures that collisions are avoided.

When the START condition has been transmitted, the master must transmit a slave address (ADDR) with an R/W bit on the bus. If this address is available in the transmit buffer, the master transmits it immediately, but if the buffer is empty, the master holds the I^2C-bus while waiting for software to write the address to the transmit buffer.

After the address has been transmitted, a sequence of bytes can be read from or written to the slave, depending on the value of the R/W bit (bit 0 in the address byte). If the bit was cleared, the master has entered a master transmitter role, where it now transmits data to the slave. If the bit was set, it has entered a master receiver role, where it now should receive data from the slave. In either case, an unlimited number of bytes can be transferred in one direction during the transmission.

At the end of the transmission, the master either transmits a repeated START condition (Sr) if it wishes to continue with another transfer, or transmits a STOP condition (P) if it wishes to release the bus. When operating in the master mode, I2CCLK frequency must be higher than 2 MHz for Standard-mode, 9 MHz for Fast-mode, and 20 MHz for Fast-mode Plus.

23.3.8.1 Master State Machine

The master state machine is shown in Figure 23.15 I2C Master State Machine on page 686. A master operation starts in the far left of the state machine, and follows the solid lines through the state machine, ending the operation or continuing with a new operation when arriving at the right side of the state machine.

Branches in the path through the state machine are the results of bus events and choices made by software, either directly or indirectly. The dotted lines show where I²C-specific interrupt flags are set along the path and the full-drawn circles show places where interaction may be required by software to let the transmission proceed.

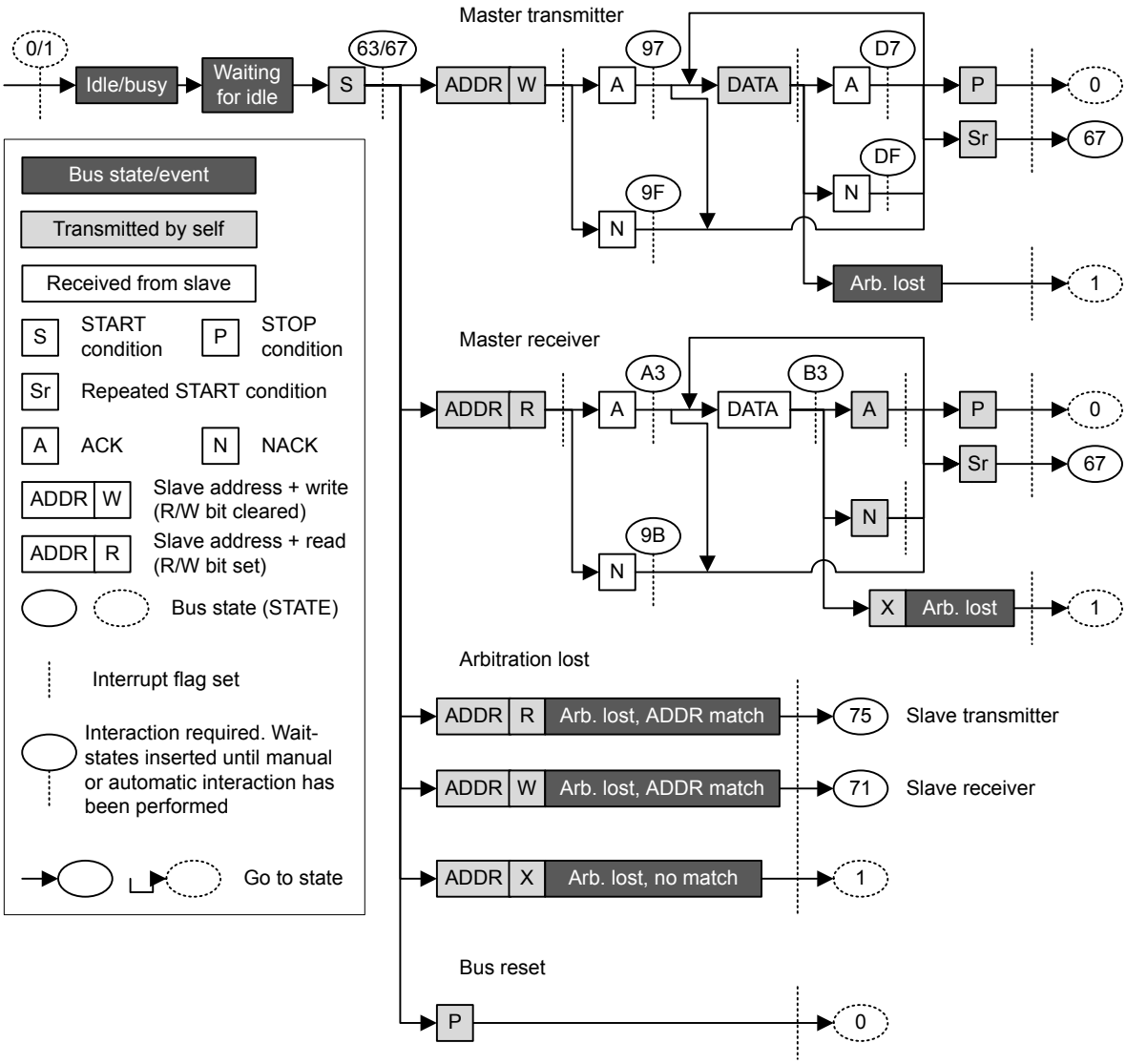


Figure 23.15. I2C Master State Machine

23.3.8.2 Interactions

Whenever the I²C module is waiting for interaction from software, it holds the bus clock SCL low, freezing all bus activities, and the BUSHOLD interrupt flag in I2C_IF is set. The action(s) required by software depends on the current state the of the I²C module. This state can be read from the I2C_STATE register.

As an example, Table 23.3 I2C Master Transmitter on page 689 shows the different states the I²C goes through when operating as a Master Transmitter, i.e., a master that transmits data to a slave. As seen in the table, when a start condition has been transmitted, a requirement is that there is an address and an R/W bit in the transmit buffer. If the transmit buffer is empty, then the BUSHOLD interrupt flag is set, and the bus is held until data becomes available in the buffer. While waiting for the address, I2C_STATE has a value 0x67, which can be used to identify exactly what the I²C module is waiting for.

Note: The bus would never stop at state 0x67 if the address was available in the transmit buffer.

The BUSHOLD interrupt flag needs to be manually cleared by software after the appropriate action has been taken.

The different interactions used by the I²C module are listed in Table 23.2 I2C Interactions in Prioritized Order on page 687 in a prioritized order. If the I²C module is in such a state that multiple courses of action are possible, then the action chosen is the one that has the highest priority. For example, after sending out a START, if an address is present in the buffer and a STOP is also pending, then the I²C will send out the STOP since it has the higher priority.

Table 23.2. I2C Interactions in Prioritized Order

Interaction	Priority	Software action	Automatically continues if
STOP*	1	Set the STOP command bit in I2C_CMD	PSTOP is set (STOP pending) in I2C_STATUS
ABORT	2	Set the ABORT command bit in I2C_CMD	Never, the transmission is aborted
CONT*	3	Set the CONT command bit in I2C_CMD	PCONT is set in I2C_STATUS (CONT pending)
NACK*	4	Set the NACK command bit in I2C_CMD	PNACK is set in I2C_STATUS (NACK pending)
ACK*	5	Set the ACK command bit in I2C_CMD	AUTOACK is set in I2C_CTRL or PACK is set in I2C_STATUS (ACK pending)
ADDR+W -> TXDATA	6	Write an address to the transmit buffer with the R/W bit set	Address is available in transmit buffer with R/W bit set
ADDR+R -> TXDATA	7	Write an address to the transmit buffer with the R/W bit cleared	Address is available in transmit buffer with R/W bit cleared
START*	8	Set the START command bit in I2C_CMD	PSTART is set in I2C_STATUS (START pending)
TXDATA/ TXDOUBLE	9	Write data to the transmit buffer	Data is available in transmit buffer
RXDATA/ RXDOUBLE	10	Read data from receive buffer	Space is available in receive buffer
None	11	No interaction is required	

The commands marked with a * in Table 23.2 I2C Interactions in Prioritized Order on page 687 can be issued before an interaction is required. When such a command is issued before it can be used/consumed by the I²C module, the command is set in a pending state, which can be read from the STATUS register. A pending START command can for instance be identified by PSTART having a high value.

Whenever the I²C module requires an interaction, it checks the pending commands. If one or a combination of these can fulfill an interaction, they are consumed by the module and the transmission continues without setting the BUSHOLD interrupt flag in I2C_IF to get an interaction from software. The pending status of a command goes low when it is consumed.

When several interactions are possible from a set of pending commands, the interaction with the highest priority, i.e., the interaction closest to the top of Table 23.2 I2C Interactions in Prioritized Order on page 687 is applied to the bus.

Pending commands can be cleared by setting the CLEARPC command bit in I2C_CMD.

23.3.8.3 Automatic ACK Interaction

When receiving addresses and data, an ACK command in I2C_CMD is normally required after each received byte. When AUTOACK is set in I2C_CTRL, an ACK is always pending, and the ACK-pending bit PACK in I2C_STATUS is thus always set, even after an ACK has been consumed. This is used when data is picked using I2C_RXDOUBLE and can also be used with I2C_RXDATA in order to reduce the amount of software interaction required during a transfer.

23.3.8.4 Reset State

After a reset, the state of the I^2C-bus is unknown. To avoid interrupting transfers on the I^2C-bus after a reset of the I^2C module or the entire MCU, the I^2C-bus is assumed to be busy when coming out of a reset, and the BUSY flag in I2C_STATUS is thus set. To be able to carry through master operations on the I^2C-bus, the bus must be idle.

The bus goes idle when a STOP condition is detected on the bus, but on buses with little activity, the time before the I^2C module detects that the bus is idle can be significant. There are two ways of assuring that the I^2C module gets out of the busy state.

- Use the ABORT command in I2C_CMD. When the ABORT command is issued, the I^2C module is instructed that the bus is idle. The I^2C module can then initiate master operations.
- Use the Bus Idle Timeout. When SCL has been high for a long period of time, it is very likely that the bus is idle. Set BITO in I2C_CTRL to an appropriate timeout period and set GIBITO in I2C_CTRL. If activity has not been detected on the bus within the timeout period, the bus is then automatically assumed idle, and master operations can be initiated.

Note: If operating in slave mode, the above approach is not necessary.

23.3.8.5 Master Transmitter

To transmit data to a slave, the master must operate as a master transmitter. Table 23.3 I2C Master Transmitter on page 689 shows the states the I²C module goes through while acting as a master transmitter. Every state where an interaction is required has the possible interactions listed, along with the result of the interactions. The table also shows which interrupt flags are set in the different states. The interrupt flags enclosed in parenthesis may be set. If the BUSHOLD interrupt in I2C_IF is set, the module is waiting for an interaction, and the bus is frozen. The value of I2C_STATE will be equal to the values given in the table when the BUSHOLD interrupt flag is set, and can be used to determine which interaction is required to make the transmission continue.

The interrupt flag START in I2C_IF is set when the I²C module transmits the START.

A master operation is started by issuing a START command by setting START in I2C_CMD. ADDR+W, i.e., the address of the slave + the R/W bit is then required by the I²C module. If this is not available in the transmit buffer, then the bus is held and the BUSHOLD interrupt flag is set. The value of I2C_STATE will then be 0x67. As seen in the table, the I²C module also stops in this state if the address is not available after a repeated start condition.

To continue, write a byte to I2C_TXDATA with the address of the slave in the 7 most significant bits and the least significant bit cleared (ADDR+W). This address will then be transmitted, and the slave will reply with an ACK or a NACK. If no slave replies to the address, the response will also be NACK. If the address was acknowledged, the master now has four choices. It can send data by placing it in I2C_TXDATA/ I2C_TXDOUBLE (the master should check the TXBL interrupt flag before writing to the transmit buffer), this data is then transmitted. The master can also stop the transmission by sending a STOP, it can send a repeated start by sending START, or it can send a STOP and then a START as soon as possible. If the master wishes to make another transfer immediately after the current, the preferred way is to start a new transfer directly by transmitting a repeated START instead of a STOP followed by a START. This is so because if a STOP is sent out, then any master wishing to initiate a transfer on the bus can try to gain control of it.

If a NACK was received, the master has to issue a CONT command in addition to providing data in order to continue transmission. This is not standard I²C, but is provided for flexibility. The rest of the options are similar to when an ACK was received.

If a new byte was transmitted, an ACK or NACK is received after the transmission of the byte, and the master has the same options as for when the address was sent.

The master may lose arbitration at any time during transmission. In this case, the ARBLOST interrupt flag in I2C_IF is set. If the arbitration was lost during the transfer of an address, and SLAVE in I2C_CTRL is set, the master then checks which address was transmitted. If it was the address of the master, then the master goes to slave mode.

After a master has transmitted a START and won any arbitration, it owns the bus until it transmits a STOP. After a STOP, the bus is released, and arbitration decides which bus master gains the bus next. The MSTOP interrupt flag in I2C_IF is set when a STOP condition is transmitted by the master.

Table 23.3. I2C Master Transmitter

I2C_STATE	Description	I2C_IF	Required interaction	Response
0x67	Start transmitted	START interrupt flag (BUSHOLD interrupt flag)	ADDR+W -> TXDATA	ADDR+W will be sent
			STOP	STOP will be sent and bus released.
			STOP + START	STOP will be sent and bus released. Then a START will be sent when bus becomes idle.
0x67	Repeated start transmitted	START interrupt flag (BUSHOLD interrupt flag)	ADDR+W -> TXDATA	ADDR+W will be sent
			STOP	STOP will be sent and bus released.
			STOP + START	STOP will be sent and bus released. Then a START will be sent when bus becomes idle.
-	ADDR+W transmitted	TXBL interrupt flag (TXC interrupt flag)	None	

I2C_STATE	Description	I2C_IF	Required interaction	Response
0x97	ADDR+W transmitted, ACK received	ACK interrupt flag (BUSHOLD interrupt flag)	TXDATA	DATA will be sent
			STOP	STOP will be sent. Bus will be released
			START	Repeated start condition will be sent
			STOP + START	STOP will be sent and the bus released. Then a START will be sent when the bus becomes idle
0x9F	ADDR+W transmitted,NACK received	NACK (BUSHOLD interrupt flag)	CONT + TXDATA	DATA will be sent
			STOP	STOP will be sent. Bus will be released
			START	Repeated start condition will be sent
			STOP + START	STOP will be sent and the bus released. Then a START will be sent when the bus becomes idle
-	Data transmitted	TXBL interrupt flag (TXC interrupt flag)	None	
0xD7	Data transmitted,ACK received	ACK interrupt flag (BUSHOLD interrupt flag)	TXDATA	DATA will be sent
			STOP	STOP will be sent. Bus will be released
			START	Repeated start condition will be sent
			STOP + START	STOP will be sent and the bus released. Then a START will be sent when the bus becomes idle
0xDF	Data transmitted,NACK received	NACK(BUSHOLD interrupt flag)	CONT + TXDATA	DATA will be sent
			STOP	STOP will be sent. Bus will be released
			START	Repeated start condition will be sent
			STOP + START	STOP will be sent and the bus released. Then a START will be sent when the bus becomes idle
-	Stop transmitted	MSTOP interrupt flag	None	
			START	START will be sent when bus becomes idle
-	Arbitration lost	ARBLOST interrupt flag	None	
			START	START will be sent when bus becomes idle

23.3.8.6 Master Receiver

To receive data from a slave, the master must operate as a master receiver, see Table 23.4 I2C Master Receiver on page 691. This is done by transmitting ADDR+R as the address byte instead of ADDR+W, which is transmitted to become a master transmitter. The address byte loaded into the data register thus has to contain the 7-bit slave address in the 7 most significant bits of the byte, and have the least significant bit set.

When the address has been transmitted, the master receives an ACK or a NACK. If an ACK is received, the ACK interrupt flag in I2C_IF is set, and if space is available in the receive shift register, reception of a byte from the slave begins. If the receive buffer and shift register is full however, the bus is held until data is read from the receive buffer or another interaction is made. Note that the STOP and START interactions have a higher priority than the data-available interaction, so if a STOP or START command is pending, the highest priority interaction will be performed, and data will not be received from the slave.

If a NACK was received, the CONT command in I2C_CMD has to be issued in order to continue receiving data, even if there is space available in the receive buffer and/or shift register.

After a data byte has been received the master must ACK or NACK the received byte. If an ACK is pending or AUTOACK in I2C_CTRL is set, an ACK is sent automatically and reception continues if space is available in the receive buffer.

If a NACK is sent, the CONT command must be used in order to continue transmission. If an ACK or NACK is issued along with a START or STOP or both, then the ACK/NACK is transmitted and the reception is ended. If START in I2C_CMD is set alone, a repeated start condition is transmitted after the ACK/NACK. If STOP in I2C_CMD is set, a stop condition is sent regardless of whether START is set. If START is set in this case, it is set as pending.

As when operating as a master transmitter, arbitration can be lost as a master receiver. When this happens the ARBLOST interrupt flag in I2C_IF is set, and the master has a possibility of being selected as a slave given the correct conditions.

Table 23.4. I2C Master Receiver

I2C_STATE	Description	I2C_IF	Required interaction	Response
0x63	START transmitted	START interrupt flag (BUSHOLD interrupt flag)	ADDR+R -> TXDATA	ADDR+R will be sent
			STOP	STOP will be sent and bus released.
			STOP + START	STOP will be sent and bus released. Then a START will be sent when bus becomes idle.
0x67	Repeated START transmitted	START interrupt flag(BUSHOLD interrupt flag)	ADDR+R -> TXDATA	ADDR+R will be sent
			STOP	STOP will be sent and bus released.
			STOP + START	STOP will be sent and bus released. Then a START will be sent when bus becomes idle.
-	ADDR+R transmitted	TXBL interrupt flag (TXC interrupt flag)	None	
0xA3	ADDR+R transmitted, ACK received	ACK interrupt flag(BUSHOLD)	RXDATA	Start receiving
			STOP	STOP will be sent and the bus released
			START	Repeated START will be sent
			STOP + START	STOP will be sent and the bus released. Then a START will be sent when the bus becomes idle
0x9B	ADDR+R transmitted,NACK received	NACK(BUSHOLD)	CONT + RXDATA	Continue, start receiving
			STOP	STOP will be sent and the bus released
			START	Repeated START will be sent
			STOP + START	STOP will be sent and the bus released. Then a START will be sent when the bus becomes idle

I2C_STATE	Description	I2C_IF	Required interaction	Response
0xB3	Data received	RXDATA interrupt flag(BUSHOLD interrupt flag)	ACK + RXDATA	ACK will be transmitted, reception continues
			NACK + CONT + RXDATA	NACK will be transmitted, reception continues
			ACK/NACK + STOP	ACK/NACK will be sent and the bus will be released.
			ACK/NACK + START	ACK/NACK will be sent, and then a repeated start condition.
			ACK/NACK + STOP + START	ACK/NACK will be sent and the bus will be released. Then a START will be sent when the bus becomes idle
-	Stop received	MSTOP interrupt flag	None	
			START	START will be sent when bus becomes idle
-	Arbitration lost	ARBLOST interrupt flag	None	
			START	START will be sent when bus becomes idle

23.3.8.7 SDA/SCL Status Monitor

The I^2C module supports an SDA and SCL monitoring function. Note that this functionality is only supported when the I2C module is in single master mode, and when the slave doesn't use clock stretching. Additionally, firmware should set the ARBDIS bit in I2C_CTRL when using the SDA/SCL monitoring to prevent the bus being released.

The SDA monitor is enabled by setting the SDAMONEN in I2C_CTRL. Once enabled, the SDA monitor will check the status of the SDA line at the following points:

- At a Start Condition, before SDA falls
- At Stop Condition, after SDA rises

After checking, the monitor will set the SDAERR flag in I2C_IF it fails to read SDA==1. To allow the SDAERR flag to generate an IRQ, set the SDAERR bit in I2C_IEN.

Similarly, the SCL monitor is enabled by setting the SCLMONEN in I2C_CTRL. Once enabled, the SCL monitor will check the status of the SCL line at the following points:

- At a Start Condition, before SCL falls
- At every clock cycle, before SCL falls
- At Stop Condition, after SCL rises

After checking, the monitor will set the SCLERR flag in I2C_IF it fails to read SCL==1. To allow the SCLERR flag to generate an IRQ, set the SCLERR bit in I2C_IEN.

23.3.9 Bus States

The I2C_STATE register can be used to determine which state the I²C module and the I²C bus are in at a given time. The register consists of the STATE bit-field, which shows which state the I²C module is at in any ongoing transmission, and a set of single-bits, which reveal the transmission mode, whether the bus is busy or idle, and whether the bus is held by this I²C module waiting for a software response.

The possible values of the STATE field are summarized in Table 23.5 I2C STATE Values on page 693. When this field is cleared, the I²C module is not a part of any ongoing transmission. The remaining status bits in the I2C_STATE register are listed in Table 23.6 I2C Transmission Status on page 693.

Table 23.5. I2C STATE Values

Mode	Value	Description
IDLE	0	No transmission is being performed by this module.
WAIT	1	Waiting for idle. Will send a start condition as soon as the bus is idle.
START	2	Start transmit phase
ADDR	3	Address transmit or receive phase
ADDRACK	4	Address ACK/NACK transmit or receive phase
DATA	5	Data transmit or receive phase
DATAACK	6	Data ACK/NACK transmit or receive phase

Table 23.6. I2C Transmission Status

Bit	Description
BUSY	Set whenever there is activity on the bus. Whether or not this module is responsible for the activity cannot be determined by this byte.
MASTER	Set when operating as a master. Cleared at all other times.
TRANSMITTER	Set when operating as a transmitter; either a master transmitter or a slave transmitter. Cleared at all other times
BUSHOLD	Set when the bus is held by this I²C module because an action is required by software.
NACK	Only valid when bus is held and STATE is ADDRACK or DATAACK. In that case it is set if a NACK was received. In all other cases, the bit is cleared.

Note: I2C_STATE reflects the internal state of the I²C module, and therefore only held constant as long as the bus is held, i.e., as long as BUSHOLD in I2C_STATUS is set.

23.3.10 Slave Operation

The I²C module operates in master mode by default. To enable slave operation, i.e., to allow the device to be addressed as an I²C slave, the SLAVE bit in I2C_CTRL must be set. In this case the I²C module operates in a mixed mode, both capable of starting transmissions as a master, and being addressed as a slave. When operating in the slave mode, I2CCLK frequency must be higher than 2 MHz for Standard-mode, 5 MHz for Fast-mode, and 14 MHz for Fast-mode Plus.

23.3.10.1 Slave State Machine

The slave state machine is shown in Figure 23.16 I2C Slave State Machine on page 694. The dotted lines show where I²C-specific interrupt flags are set. The full-drawn circles show places where interaction may be required by software to let the transmission proceed.

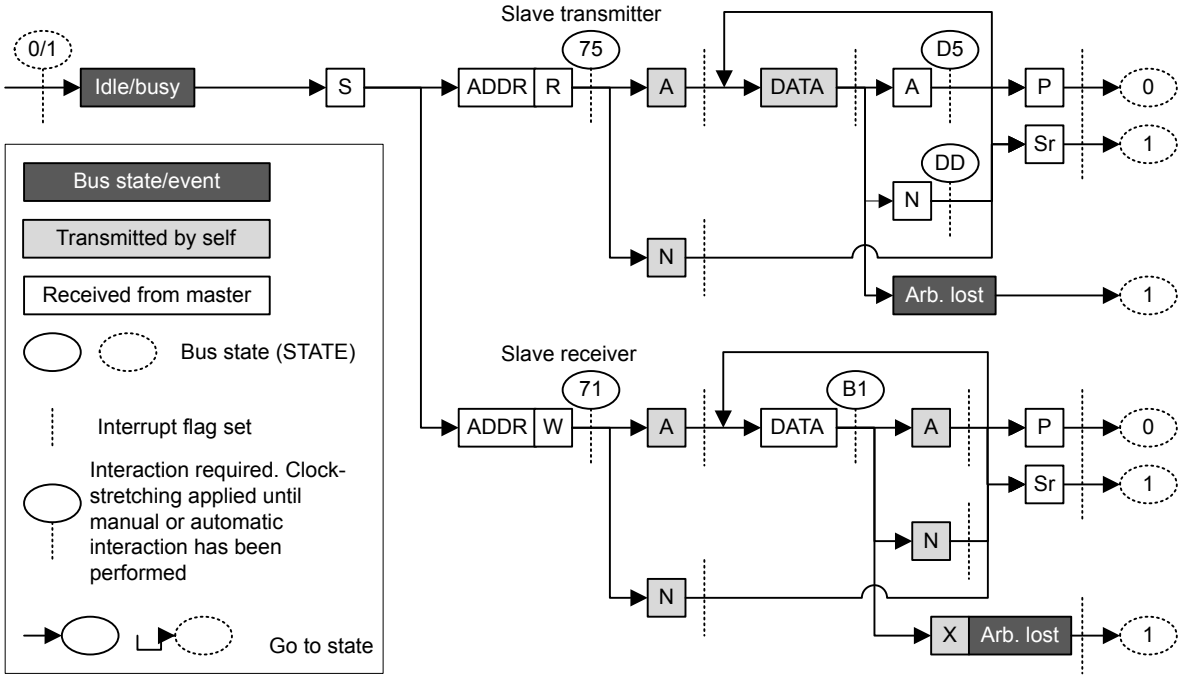


Figure 23.16. I2C Slave State Machine

23.3.10.2 Address Recognition

The I²C module provides automatic address recognition for 7-bit addresses. 10-bit address recognition is not fully automatic, but can be assisted by the 7-bit address comparator as shown in 23.3.12 Using 10-bit Addresses. Address recognition is supported in EM2/3 for I2C0 - however, the I²C0 module must be configured to use pins on either Port A or B if wakeup on address recognition from EM2/3 is desired. All other ports are available only in EM0/1. See GPIO chapter for more details.

The slave address, i.e., the address which the I²C module should be addressed with, is defined in the I2C_SADDR register. In addition to the address, a mask must be specified, telling the address comparator which bits of an incoming address to compare with the address defined in I2C_SADDR. The mask is defined in I2C_SADDRMASK, and for every zero in the mask, the corresponding bit in the slave address is treated as a don't-care, i.e., the 0-masked bits are ignored.

An incoming address that fails address recognition is automatically replied to with a NACK. Since only the bits defined by the mask are checked, a mask with a value 0x00 will result in all addresses being accepted. A mask with a value 0x7F will only match the exact address defined in I2C_SADDR, while a mask 0x70 will match all addresses where the three most significant bits in I2C_SADDR and the incoming address are equal.

If GCAMEN in I2C_CTRL is not set, the start-byte, i.e., the general call address with the R/W bit set is ignored unless it is included in the defined slave address and the address mask.

When an address is accepted by the address comparator, the decision of whether to ACK or NACK the address is passed to software.

23.3.10.3 Slave Transmitter

When SLAVE in I2C_CTRL is set, the RSTART interrupt flag in I2C_IF will be set when repeated START conditions are detected. After a START or repeated START condition, the bus master will transmit an address along with an R/W bit. If there is no room in the receive shift register for the address, the bus will be held by the slave until room is available in the shift register. Transmission then continues and the address is loaded into the shift register. If this address does not pass address recognition, it is automatically NACK'ed by the slave, and the slave goes to an idle state. The address byte is in this case discarded, making the shift register ready for a new address. It is not loaded into the receive buffer.

If the address was accepted and the R/W bit was set (R), indicating that the master wishes to read from the slave, the slave now goes into the slave transmitter mode. Software interaction is now required to decide whether the slave wants to acknowledge the request or not. The accepted address byte is loaded into the receive buffer like a regular data byte. If no valid interaction is pending, the bus is held until the slave responds with a command. The slave can reject the request with a single NACK command.

The slave will in that case go to an idle state, and wait for the next start condition. To continue the transmission, the slave must make sure data is loaded into the transmit buffer and send an ACK. The loaded data will then be transmitted to the master, and an ACK or NACK will be received from the master.

Data transmission can also continue after a NACK if a CONT command is issued along with the NACK. This is not standard I^2C however.

If the master responds with an ACK, it may expect another byte of data, and data should be made available in the transmit buffer. If data is not available, the bus is held until data is available.

If the response is a NACK however, this is an indication of that the master has received enough bytes and wishes to end the transmission. The slave now automatically goes idle, unless CONT in I2C_CMD is set and data is available for transmission. The latter is not standard I^2C.

The master ends the transmission by sending a STOP or a repeated START. The SSTOP interrupt flag in I2C_IF is set when the master transmits a STOP condition. If the transmission is ended with a repeated START, then the SSTOP interrupt flag is not set.

Note: The SSTOP interrupt flag in I2C_IF will be set regardless of whether the slave is participating in the transmission or not, as long as SLAVE in I2C_CTRL is set and a STOP condition is detected

If arbitration is lost at any time during transmission, the ARBLOST interrupt flag in I2C_IF is set, the bus is released and the slave goes idle.

See Table 23.7 I2C Slave Transmitter on page 695 for more information.

Table 23.7. I2C Slave Transmitter

I2C_STATE	Description	I2C_IF	Required interaction	Response
0x01	Repeated START received	RSTART interrupt flag (BUSHOLD interrupt flag)	RXDATA	Receive and compare address
0x75	ADDR + R received	ADDR interrupt flag	ACK + TXDATA	ACK will be sent, then DATA
		RXDATA interrupt flag	NACK	NACK will be sent, slave goes idle
		(BUSHOLD interrupt flag)	NACK + CONT + TXDATA	NACK will be sent, then DATA.
-	Data transmitted	TXBL interrupt flag (TXC interrupt flag)	None	
0xD5	Data transmitted, ACK received	ACK interrupt flag (BUSHOLD interrupt flag)	TXDATA	DATA will be transmitted
0xDD	Data transmitted, NACK received	NACK interrupt flag	None	The slave goes idle
		(BUSHOLD interrupt flag)	CONT + TXDATA	DATA will be transmitted

I2C_STATE	Description	I2C_IF	Required interaction	Response
-	Stop received	SSTOP interrupt flag	None	The slave goes idle
			START	START will be sent when bus becomes idle
-	Arbitration lost	ARBLOST interrupt flag	None	The slave goes idle
			START	START will be sent when the bus becomes idle

23.3.10.4 Slave Receiver

A slave receiver operation is started in the same way as a slave transmitter operation, with the exception that the address transmitted by the master has the R/W bit cleared (W), indicating that the master wishes to write to the slave. The slave then goes into slave receiver mode.

To receive data from the master, the slave should respond to the address with an ACK and make sure space is available in the receive buffer. Transmission will then continue, and the slave will receive a byte from the master.

If a NACK is sent without a CONT, the transmission is ended for the slave, and it goes idle. If the slave issues both the NACK and CONT commands and has space available in the receive buffer, it will be open for continuing reception from the master.

When a byte has been received from the master, the slave must ACK or NACK the byte. The responses here are the same as for the reception of the address byte.

The master ends the transmission by sending a STOP or a repeated START. The SSTOP interrupt flag is set when the master transmits a STOP condition. If the transmission is ended with a repeated START, then the SSTOP interrupt flag in I2C_IF is not set.

Note: The SSTOP interrupt flag in I2C_IF will be set regardless of whether the slave is participating in the transmission or not, as long as SLAVE in I2C_CTRL is set and a STOP condition is detected

If arbitration is lost at any time during transmission, the ARBLOST interrupt flag in I2C_IF is set, the bus is released and the slave goes idle.

See Table 23.8 I2C - Slave Receiver on page 697 for more information.

Table 23.8. I2C - Slave Receiver

I2C_STATE	Description	I2C_IF	Required interaction	Response
0x01	Repeated START received	RSTART interrupt flag (BUSHOLD interrupt flag)	RXDATA	Receive and compare address
0x71	ADDR + W received	ADDR interrupt flag RXDATA interrupt flag (BUSHOLD interrupt flag)	ACK + RXDATA	ACK will be sent and data will be received
			NACK	NACK will be sent, slave goes idle
			NACK + CONT + RXDATA	NACK will be sent and DATA will be received.
0xB1	Data received	RXDATA interrupt flag (BUSHOLD interrupt flag)	ACK + RXDATA	ACK will be sent and data will be received
			NACK	NACK will be sent and slave will go idle
			NACK + CONT + RXDATA	NACK will be sent and data will be received
-	Stop received	SSTOP interrupt flag	None	The slave goes idle
			START	START will be sent when bus becomes idle
-	Arbitration lost	ARBLOST interrupt flag	None	The slave goes idle
			START	START will be sent when the bus becomes idle

23.3.11 Transfer Automation

The I^2C can be set up to complete transfers with a minimal amount of interaction.

23.3.11.1 DMA

DMA can be used to automatically load data into the transmit buffer and load data out from the receive buffer. When using DMA, software is thus relieved of moving data to and from memory after each transferred byte.

23.3.11.2 Automatic ACK

When AUTOACK in I2C_CTRL is set, an ACK is sent automatically whenever an ACK interaction is possible and no higher priority interactions are pending.

23.3.11.3 Automatic STOP

A STOP can be generated automatically on two conditions. These apply only to the master transmitter.

If AUTOSN in I2C_CTRL is set, the I^{2}C module ends a transmission by transmitting a STOP condition when operating as a master transmitter and a NACK is received.

If AUTOSE in I2C_CTRL is set, the I^{2}C module always ends a transmission when there is no more data in the transmit buffer. If data has been transmitted on the bus, the transmission is ended after the (N)ACK has been received by the slave. If a START is sent when no data is available in the transmit buffer and AUTOSE is set, then the STOP condition is sent immediately following the START. Software must thus make sure data is available in the transmit buffer before the START condition has been fully transmitted if data is to be transferred.

23.3.12 Using 10-bit Addresses

When using 10-bit addresses in slave mode, set the I2C_SADDR register to 1111 0XX where XX are the two most significant bits of the 10-bit address, and set I2C_SADDRMASK to 0xFF. Address matches will now be given on all 10-bit addresses where the two most significant bits are correct.

When receiving an address match, the slave must acknowledge the address and receive the first data byte. This byte contains the second part of the 10-bit address. If it matches the address of the slave, the slave should ACK the byte to continue the transmission, and if it does not match, the slave should NACK it.

When the master is operating as a master transmitter, the data bytes will follow after the second address byte. When the master is operating as a master receiver however, a repeated START condition is sent after the second address byte. The address sent after this repeated START is equal to the first of the address bytes transmitted previously, but now with the R/W byte set, and only the slave that found a match on the entire 10-bit address in the previous message should ACK this address. The repeated start should take the master into a master receiver mode, and after the single address byte sent this time around, the slave begins transmission to the master.

23.3.13 Error Handling

Note: Some registers in the I^{2}C module are considered static. This means that these need to be set before an I^{2}C transaction starts and need to stay stable during the entire transaction.

Specifically:

- The GCAMEN and SLAVE fields in the I2C_CTRL register
- The I2C_SADDR register
- The GPIO_DBUSI2Cn_ROUTEEN, GPIO_DBUSI2Cn_SCLROUTE, and GPIO_DBUSI2Cn_SDAROUTE registers

23.3.13.1 ABORT Command

Some bus errors may require software intervention to be resolved. The I^{2}C module provides an ABORT command, which can be set in I2C_CMD, to help resolve bus errors.

When the bus for some reason is locked up and the I^{2}C module is in the middle of a transmission it cannot get out of, or for some other reason the I^{2}C wants to abort a transmission, the ABORT command can be used.

Setting the ABORT command will make the I^{2}C module discard any data currently being transmitted or received, release the SDA and SCL lines and go to an idle mode. ABORT effectively makes the I^{2}C module forget about any ongoing transfers.

23.3.13.2 Bus Reset

A bus reset can be performed by setting the START and STOP commands in I2C_CMD while the transmit buffer is empty. A START condition will then be transmitted, immediately followed by a STOP condition. A bus reset can also be performed by transmitting a START command with the transmit buffer empty and AUTOSE set.

23.3.13.3 I2C-Bus Errors

An I^2C-bus error occurs when a START or STOP condition is misplaced, which happens when the value on SDA changes while SCL is high during bit-transmission on the I^2C-bus. If the I^2C module is part of the current transmission when a bus error occurs, any data currently being transmitted or received is discarded, SDA and SCL are released, the BUSERR interrupt flag in I2C_IF is set to indicate the error, and the module automatically takes a course of action as defined in Table 23.9 I2C Bus Error Response on page 699.

Table 23.9. I2C Bus Error Response

	Misplaced START	Misplaced STOP
In a master/slave operation	Treated as START. Receive address.	Go idle. Perform any pending actions.

23.3.13.4 Bus Lockup

A lockup occurs when a master or slave on the I^2C-bus has locked the SDA or SCL at a low value, preventing other devices from putting high values on the bus, and thus making communication on the bus impossible.

Many slave-only devices operating on an I^2C-bus are not capable of driving SCL low, but in the rare case that SCL is stuck LOW, the advice is to apply a hardware reset signal to the slaves on the bus. If this does not work, cycle the power to the devices in order to make them release SCL.

When SDA is stuck low and SCL is free, a master should send 9 clock pulses on SCL while tristating the SDA. This procedure is performed in the GPIO module after clearing the GPIO_DBUSI2Cn_ROUTEEN register and disabling the I2C module. The device that held the bus low should release it sometime within those 9 clocks. If not, use the same approach as for when SCL is stuck, resetting and possibly cycling power to the slaves.

Lockup of SDA can be detected by keeping count of the number of continuous arbitration losses during address transmission. If arbitration is also lost during the transmission of a general call address, i.e., during the transmission of the STOP condition, which should never happen during normal operation, this is a good indication of SDA lockup.

Detection of SCL lockups can be done using the timeout functionality defined in 23.3.13.6 Clock Low Timeout

23.3.13.5 Bus Idle Timeout

When SCL has been high for a significant amount of time, this is a good indication of that the bus is idle. On an SMBus system, the bus is only allowed to be in this state for a maximum of 50 µs before the bus is considered idle.

The bus idle timeout BITO in I2C_CTRL can be used to detect situations where the bus goes idle in the middle of a transmission. The timeout can be configured in BITO, and when the bus has been idle for the given amount of time, the BITO interrupt flag in I2C_IF is set. The bus can also be set idle automatically on a bus idle timeout. This is enabled by setting GIBITO in I2C_CTRL.

When the bus idle timer times out, it wraps around and continues counting as long as its condition is true. If the bus is not set idle using GIBITO or the ABORT command in I2C_CMD, this will result in periodic timeouts.

Note: This timeout will be generated even if SDA is held low.

The bus idle timeout is active as long as the bus is busy, i.e., BUSY in I2C_STATUS is set. The timeout can be used to get the I^2C module out of the busy-state it enters when reset, see 23.3.8.4 Reset State.

23.3.13.6 Clock Low Timeout

The clock timeout, which can be configured in CLTO in I2C_CTRL, starts counting whenever SCL goes low, and times out if SCL does not go high within the configured timeout. A clock low timeout results in CLTOIF in I2C_IF being set, allowing software to take action.

When the timer times out, it wraps around and continues counting as long as SCL is low. An SCL lockup will thus result in periodic clock low timeouts as long as SCL is low.

23.3.13.7 Clock Low Error

The I²C module can continue transmission in parallel with another device for the entire transaction, as long as the two communications are identical. A case may arise when (before an arbitration has been decided upon) the I²C module decides to send out a repeated START or a STOP condition while the other device is still sending data. In the I²C protocol specifications, such a combination results in an undefined condition. The I²C deals with this by generating a clock low error. This means that if the I²C is transmitting a repeated START or a STOP condition and another device (another master or a misbehaving slave) pulls SCL low before the I²C sends out the START/STOP condition on SDA, a clock low error is generated. The CLERR interrupt flag is then set in the I2C_IF register, any held lines are released and the I²C device goes to idle.

23.3.14 DMA Support

The I²C module has full DMA support. A request for the DMA controller to write to the I²C transmit buffer can come from TXBL (transmit buffer has room for more data). The DMA controller can write to the transmit buffer using the I2C_TXDATA or the I2C_TXDOUBLE register. In order to write to the I2C_TXDOUBLE register (i.e., transferring 2 bytes simultaneously to the transmit buffer using the DMA), DMA_USEBURSTS needs to be set to 1 for the selected DMA channel. This ensures that the transfer is made to the transmit buffer only when both buffer elements are empty. For performing a DMA write to the I2C_TXDATA register, DMA_USEBURSTC needs to be set to 1 for the selected DMA channel. This ensures that a DMA transfer is made even when the transmit buffer is half-empty.

A request for the DMA controller to read from the I²C receive buffer can come from RXDATAV (data available in the receive buffer). To receive from I2C_RXDOUBLE (i.e., receive only when both buffer elements are full), DMA_USEBURSTS needs to be set to 1 for the selected DMA channel. In order to receive from I2C_RXDATA through the DMA, DMA_USEBURSTC needs to be set to 1. This ensures that the data gets picked up even when the receive buffer is half-full.

23.3.15 Interrupts

The interrupts generated by the I²C module are combined into one interrupt vector, I2C_INT. If I²C interrupts are enabled, an interrupt will be made if one or more of the interrupt flags in I2C_IF and their corresponding bits in I2C_IEN are set.

23.3.16 Wake-up

The I²C receive section can be active all the way down to energy mode EM3 stop, and can wake up the CPU on address interrupt. All address match modes are supported.

23.4 I2C Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	I2C_IPVERSION	R	IP VERSION Register
0x004	I2C_EN	RW	Enable Register
0x008	I2C_CTRL	RW	Control Register
0x00C	I2C_CMD	W	Command Register
0x010	I2C_STATE	RH	State Register
0x014	I2C_STATUS	RH	Status Register
0x018	I2C_CLKDIV	RW	Clock Division Register
0x01C	I2C_SADDR	RW	Slave Address Register
0x020	I2C_SADDRMASK	RW	Slave Address Mask Register
0x024	I2C_RXDATA	RH	Receive Buffer Data Register
0x028	I2C_RXDOUBLE	RH	Receive Buffer Double Data Register
0x02C	I2C_RXDATAP	RH	Receive Buffer Data Peek Register
0x030	I2C_RXDOUBLEP	RH	Receive Buffer Double Data Peek Register
0x034	I2C_TXDATA	W	Transmit Buffer Data Register
0x038	I2C_TXDOUBLE	W	Transmit Buffer Double Data Register
0x03C	I2C_IF	RWH INTFLAG	Interrupt Flag Register
0x040	I2C_IEN	RW	Interrupt Enable Register
0x1000	I2C_IPVERSION_SET	R	IP VERSION Register
0x1004	I2C_EN_SET	RW	Enable Register
0x1008	I2C_CTRL_SET	RW	Control Register
0x100C	I2C_CMD_SET	W	Command Register
0x1010	I2C_STATE_SET	RH	State Register
0x1014	I2C_STATUS_SET	RH	Status Register
0x1018	I2C_CLKDIV_SET	RW	Clock Division Register
0x101C	I2C_SADDR_SET	RW	Slave Address Register
0x1020	I2C_SADDRMASK_SET	RW	Slave Address Mask Register
0x1024	I2C_RXDATA_SET	RH	Receive Buffer Data Register
0x1028	I2C_RXDOUBLE_SET	RH	Receive Buffer Double Data Register
0x102C	I2C_RXDATAP_SET	RH	Receive Buffer Data Peek Register
0x1030	I2C_RXDOUBLEP_SET	RH	Receive Buffer Double Data Peek Register
0x1034	I2C_TXDATA_SET	W	Transmit Buffer Data Register
0x1038	I2C_TXDOUBLE_SET	W	Transmit Buffer Double Data Register
0x103C	I2C_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1040	I2C_IEN_SET	RW	Interrupt Enable Register
0x2000	I2C_IPVERSION_CLR	R	IP VERSION Register

Offset	Name	Type	Description
0x2004	I2C_EN_CLR	RW	Enable Register
0x2008	I2C_CTRL_CLR	RW	Control Register
0x200C	I2C_CMD_CLR	W	Command Register
0x2010	I2C_STATE_CLR	RH	State Register
0x2014	I2C_STATUS_CLR	RH	Status Register
0x2018	I2C_CLKDIV_CLR	RW	Clock Division Register
0x201C	I2C_SADDR_CLR	RW	Slave Address Register
0x2020	I2C_SADDRMASK_CLR	RW	Slave Address Mask Register
0x2024	I2C_RXDATA_CLR	RH	Receive Buffer Data Register
0x2028	I2C_RXDOUBLE_CLR	RH	Receive Buffer Double Data Register
0x202C	I2C_RXDATAP_CLR	RH	Receive Buffer Data Peek Register
0x2030	I2C_RXDOUBLEP_CLR	RH	Receive Buffer Double Data Peek Register
0x2034	I2C_TXDATA_CLR	W	Transmit Buffer Data Register
0x2038	I2C_TXDOUBLE_CLR	W	Transmit Buffer Double Data Register
0x203C	I2C_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2040	I2C_IEN_CLR	RW	Interrupt Enable Register
0x3000	I2C_IPVERSION_TGL	R	IP VERSION Register
0x3004	I2C_EN_TGL	RW	Enable Register
0x3008	I2C_CTRL_TGL	RW	Control Register
0x300C	I2C_CMD_TGL	W	Command Register
0x3010	I2C_STATE_TGL	RH	State Register
0x3014	I2C_STATUS_TGL	RH	Status Register
0x3018	I2C_CLKDIV_TGL	RW	Clock Division Register
0x301C	I2C_SADDR_TGL	RW	Slave Address Register
0x3020	I2C_SADDRMASK_TGL	RW	Slave Address Mask Register
0x3024	I2C_RXDATA_TGL	RH	Receive Buffer Data Register
0x3028	I2C_RXDOUBLE_TGL	RH	Receive Buffer Double Data Register
0x302C	I2C_RXDATAP_TGL	RH	Receive Buffer Data Peek Register
0x3030	I2C_RXDOUBLEP_TGL	RH	Receive Buffer Double Data Peek Register
0x3034	I2C_TXDATA_TGL	W	Transmit Buffer Data Register
0x3038	I2C_TXDOUBLE_TGL	W	Transmit Buffer Double Data Register
0x303C	I2C_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3040	I2C_IEN_TGL	RW	Interrupt Enable Register

23.5 I2C Register Description

23.5.1 I2C_IPVERSION - IP VERSION Register

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

23.5.2 I2C_EN - Enable Register

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**module enable**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

Value	Mode	Description
0	DISABLE	Disable Peripheral Clock
1	ENABLE	Enable Peripheral Clock

23.5.3 I2C_CTRL - Control Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x0	0x0		0x0			0x0		0x0				0x0		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access											RW	RW		RW			RW		RW				RW		RW	RW	RW	RW	RW	RW	RW	RW
Name											SDAMONEN	SCLMONEN		CLTO			GIBITO		BITO				CLHR		TXBIL	GCAMEN	ARBDIS	AUTOSN	AUTOSE	AUTOACK	SLAVE	CORERST

Bit	Name	Reset	Access	Description
31:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	SDAMONEN	0x0	RW	**SDA Monitor Enable**

Set to enable SDA monitor feature. This will enable SDA rise check at loopback path. This monitor can not be enabled in MultiMaster application

Value	Mode	Description
0	DISABLE	Disable SDA Monitor
1	ENABLE	Enable SDA Monitor

20	SCLMONEN	0x0	RW	**SCL Monitor Enable**

Set to enable SCL monitor feature. This will enable SCL rise check at loopback path. This monitor can not be enabled in MultiMaster application

Value	Mode	Description
0	DISABLE	Disable SCL monitor
1	ENABLE	Enable SCL monitor

19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	CLTO	0x0	RW	**Clock Low Timeout**

Use to generate a timeout when CLK has been low for the given amount of time. Wraps around and continues counting when the timeout is reached. The timeout value can be calculated by timeout = PCC/(Fscl x (Nlow + Nhigh))

Value	Mode	Description
0	OFF	Timeout disabled
1	I2C40PCC	Timeout after 40 prescaled clock cycles. In standard mode at 100 kHz, this results in a 50us timeout.
2	I2C80PCC	Timeout after 80 prescaled clock cycles. In standard mode at 100 kHz, this results in a 100us timeout.
3	I2C160PCC	Timeout after 160 prescaled clock cycles. In standard mode at 100 kHz, this results in a 200us timeout.
4	I2C320PCC	Timeout after 320 prescaled clock cycles. In standard mode at 100 kHz, this results in a 400us timeout.

Bit	Name	Reset	Access	Description
	5	I2C1024PCC		Timeout after 1024 prescaled clock cycles. In standard mode at 100 kHz, this results in a 1280us timeout.
15	GIBITO	0x0	RW	**Go Idle on Bus Idle Timeout**
	When set, the bus automatically goes idle on a bus idle timeout, allowing new transfers to be initiated.			
	Value	Mode		Description
	0	DISABLE		A bus idle timeout has no effect on the bus state.
	1	ENABLE		A bus idle timeout tells the I2C module that the bus is idle, allowing new transfers to be initiated.
14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	BITO	0x0	RW	**Bus Idle Timeout**
	Use to generate a timeout when SCL has been high for a given amount time between a START and STOP condition. When in a bus transaction, i.e. the BUSY flag is set, a timer is started whenever SCL goes high. When the timer reaches the value defined by BITO, it sets the BITO interrupt flag. The BITO interrupt flag will then be set periodically as long as SCL remains high. The bus idle timeout is active as long as BUSY is set. It is thus stopped automatically on a timeout if GIBITO is set. It is also stopped a STOP condition is detected and when the ABORT command is issued. The timeout is activated whenever the bus goes BUSY, i.e. a START condition is detected. The timeout value can be calculated by timeout = PCC/(Fscl x (Nlow + Nhigh))			
	Value	Mode		Description
	0	OFF		Timeout disabled
	1	I2C40PCC		Timeout after 40 prescaled clock cycles. In standard mode at 100 kHz, this results in a 50us timeout.
	2	I2C80PCC		Timeout after 80 prescaled clock cycles. In standard mode at 100 kHz, this results in a 100us timeout.
	3	I2C160PCC		Timeout after 160 prescaled clock cycles. In standard mode at 100 kHz, this results in a 200us timeout.
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	CLHR	0x0	RW	**Clock Low High Ratio**
	Determines the values of (and ratio between) the low and high parts of the clock signal generated on SCL as master.			
	Value	Mode		Description
	0	STANDARD		Nlow=4 and Nhigh=4, and the Nlow:Nhigh ratio is 4:4
	1	ASYMMETRIC		Nlow=6 and Nhigh=3, and the Nlow:Nhigh ratio is 6:3
	2	FAST		Nlow=11 and Nhigh=6, and the Nlow:Nhigh ratio is 11:6
7	TXBIL	0x0	RW	**TX Buffer Interrupt Level**
	Determines the interrupt and status level of the transmit buffer.			
	Value	Mode		Description
	0	EMPTY		TXBL status and the TXBL interrupt flag are set when the transmit buffer becomes empty. TXBL is cleared when the buffer becomes nonempty.

Bit	Name	Reset	Access	Description
	1	HALF_FULL		TXBL status and the TXBL interrupt flag are set when the transmit buffer goes from full to half-full or empty. TXBL is cleared when the buffer becomes full
6	GCAMEN	0x0	RW	**General Call Address Match Enable**
	Set to enable address match on general call in addition to the programmed slave address.			
	Value	Mode		Description
	0	DISABLE		General call address will be NACK'ed if it is not included by the slave address and address mask.
	1	ENABLE		When a general call address is received, a software response is required
5	ARBDIS	0x0	RW	**Arbitration Disable**
	A master or slave will not release the bus upon losing arbitration.			
	Value	Mode		Description
	0	DISABLE		When a device loses arbitration, the ARBIF interrupt flag is set and the bus is released.
	1	ENABLE		When a device loses arbitration, the ARBIF interrupt flag is set, but communication proceeds.
4	AUTOSN	0x0	RW	**Automatic STOP on NACK**
	Write to 1 to make a master transmitter send a STOP when a NACK is received from a slave.			
	Value	Mode		Description
	0	DISABLE		Stop is not automatically sent if a NACK is received from a slave.
	1	ENABLE		The master automatically sends a STOP if a NACK is received from a slave.
3	AUTOSE	0x0	RW	**Automatic STOP when Empty**
	Write to 1 to make a master transmitter send a STOP when no more data is available for transmission.			
	Value	Mode		Description
	0	DISABLE		A stop must be sent manually when no more data is to be transmitted.
	1	ENABLE		The master automatically sends a STOP when no more data is available for transmission.
2	AUTOACK	0x0	RW	**Automatic Acknowledge**
	Set to enable automatic acknowledges.			
	Value	Mode		Description
	0	DISABLE		Software must give one ACK command for each ACK transmitted on the I2C bus.
	1	ENABLE		Addresses that are not automatically NACK'ed, and all data is automatically acknowledged.
1	SLAVE	0x0	RW	**Addressable as Slave**

Bit	Name	Reset	Access	Description
	Set this bit to allow the device to be selected as an I2C slave.			

Value	Mode	Description
0	DISABLE	All addresses will be responded to with a NACK
1	ENABLE	Addresses matching the programmed slave address or the general call address (if enabled) require a response from software. Other addresses are automatically responded to with a NACK.

Bit	Name	Reset	Access	Description
0	CORERST	0x0	RW	**Soft Reset the internal state registers**
	Set to reset the I2C_STATE register, and return the I2C module to the IDLE state. Must clear this bit to resume normal operation condition			

Value	Mode	Description
0	DISABLE	No change to internal state registers
1	ENABLE	Reset the internal state registers

23.5.4 I2C_CMD - Command Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																									W	W	W	W	W	W	W	W
Name																									CLEARPC	CLEARTX	ABORT	CONT	NACK	ACK	STOP	START

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	CLEARPC	0x0	W	**Clear Pending Commands**
	Set to clear pending commands.			
6	CLEARTX	0x0	W	**Clear TX**
	Set to clear transmit buffer and shift register. Will not abort ongoing transfer.			
5	ABORT	0x0	W	**Abort transmission**
	Abort the current transmission making the bus go idle. When used in combination with STOP, a STOP condition is sent as soon as possible before aborting the transmission. The stop condition is subject to clock synchronization.			
4	CONT	0x0	W	**Continue transmission**
	Set to continue transmission after a NACK has been received.			
3	NACK	0x0	W	**Send NACK**
	Set to transmit a NACK the next time an acknowledge is required.			
2	ACK	0x0	W	**Send ACK**
	Set to transmit an ACK the next time an acknowledge is required.			
1	STOP	0x0	W	**Send stop condition**
	Set to send stop condition as soon as possible.			
0	START	0x0	W	**Send start condition**
	Set to send start condition as soon as possible. If a transmission is ongoing and not owned, the start condition will be sent as soon as the bus is idle. If the current transmission is owned by this module, a repeated start condition will be sent. Use in combination with a STOP command to automatically send a STOP, then a START when the bus becomes idle.			

23.5.5 I2C_STATE - State Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0			0x0	0x0	0x0	0x0	0x1
Access																									R			R	R	R	R	R
Name																									STATE			BUSHOLD	NACKED	TRANSMITTER	MASTER	BUSY

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:5	STATE	0x0	R	**Transmission State**
	The state of any current transmission. Cleared if the I2C module is idle.			

Value	Mode	Description
0	IDLE	No transmission is being performed.
1	WAIT	Waiting for idle. Will send a start condition as soon as the bus is idle.
2	START	Start transmit phase
3	ADDR	Address transmit or receive phase
4	ADDRACK	Address ack/nack transmit or receive phase
5	DATA	Data transmit or receive phase
6	DATAACK	Data ack/nack transmit or receive phase

4	BUSHOLD	0x0	R	**Bus Held**
	Set if the bus is currently being held by this I2C module.			
3	NACKED	0x0	R	**Nack Received**
	Set if a NACK was received and STATE is ADDRACK or DATAACK.			
2	TRANSMITTER	0x0	R	**Transmitter**
	Set when operating as a master transmitter or a slave transmitter. When cleared, the system may be operating as a master receiver, a slave receiver or the current mode is not known.			
1	MASTER	0x0	R	**Master**
	Set when operating as an I2C master. When cleared, the system may be operating as an I2C slave.			
0	BUSY	0x1	R	**Bus Busy**
	Set when the bus is busy. Whether the I2C module is in control of the bus or not has no effect on the value of this bit. When the MCU comes out of reset, the state of the bus is not known, and thus BUSY is set. Use the ABORT command or a bus idle timeout to force the I2C module out of the BUSY state.			

23.5.6 I2C_STATUS - Status Register

Offset	Bit Position																											
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0x0		0x0	0x0	0x1	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																					R		R	R	R	R	R	R	R	R	R	R
Name																					TXBUFCNT		RXFULL	RXDATAV	TXBL	TXC	PABORT	PCONT	PNACK	PACK	PSTOP	PSTART

Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:10	TXBUFCNT	0x0	R	**TX Buffer Count** Indicates the number of buffers filled with valid data and not transmit to tx shift register
9	RXFULL	0x0	R	**RX FIFO Full** Set when the receive buffer is full. Cleared when the receive buffer is no longer full. When this bit is set, there is still room for one more frame in the receive shift register.
8	RXDATAV	0x0	R	**RX Data Valid** Set when data is available in the receive buffer. Cleared when the receive buffer is empty.
7	TXBL	0x1	R	**TX Buffer Level** Indicates the level of the transmit buffer. if TXBIL==0, set when the transmit buffer is empty. if TXBIL==1, set when the transmit buffer is half full
6	TXC	0x0	R	**TX Complete** Set when a transmission has completed and no more data is available in the transmit buffer. Cleared when a new transmission starts.
5	PABORT	0x0	R	**Pending abort** An abort is pending and will be transmitted as soon as possible.
4	PCONT	0x0	R	**Pending continue** A continue is pending and will be transmitted as soon as possible.
3	PNACK	0x0	R	**Pending NACK** A not-acknowledge is pending and will be transmitted as soon as possible.
2	PACK	0x0	R	**Pending ACK** An acknowledge is pending and will be transmitted as soon as possible.
1	PSTOP	0x0	R	**Pending STOP** A stop condition is pending and will be transmitted as soon as possible.
0	PSTART	0x0	R	**Pending START** A start condition is pending and will be transmitted as soon as possible.

23.5.7 I2C_CLKDIV - Clock Division Register

Offset	Bit Position																															
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												RW				
Name																												DIV				

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	DIV	0x0	RW	**Clock Divider**
	Specifies the clock divider for the I2C. Note that DIV must be 1 or higher when slave is enabled.			

23.5.8 I2C_SADDR - Slave Address Register

Offset	Bit Position																															
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												RW				
Name																												ADDR				

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:1	ADDR	0x0	RW	**Slave address**
	Specifies the slave address of the device.			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

23.5.9 I2C_SADDRMASK - Slave Address Mask Register

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												RW				
Name																												SADDRMASK				

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:1	SADDRMASK	0x0	RW	**Slave Address Mask**
	Specifies the significant bits of the slave address. Setting the mask to 0x00 will match all addresses, while setting it to 0x7F will only match the exact address specified by ADDR.			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

23.5.10 I2C_RXDATA - Receive Buffer Data Register

Offset	Bit Position																															
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												R				
Name																												RXDATA				

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	RXDATA	0x0	R	**RX Data**
	Use this register to read from the receive buffer. Buffer is emptied on read access.			

23.5.11 I2C_RXDOUBLE - Receive Buffer Double Data Register

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0								0x0							
Access																	R								R							
Name																	RXDATA1								RXDATA0							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	RXDATA1	0x0	R	**RX Data 1**
	Second byte read from buffer. Buffer is emptied on read access.			
7:0	RXDATA0	0x0	R	**RX Data 0**
	First byte read from buffer. Buffer is emptied on read access.			

23.5.12 I2C_RXDATAP - Receive Buffer Data Peek Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									RXDATAP							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	RXDATAP	0x0	R	**RX Data Peek**
	Use this register to read from the receive buffer. Buffer is not emptied on read access.			

23.5.13 I2C_RXDOUBLEP - Receive Buffer Double Data Peek Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0								0x0							
Access																	R								R							
Name																	RXDATAP1								RXDATAP0							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	RXDATAP1	0x0	R	**RX Data 1 Peek**
	Second byte read from buffer. Buffer is not emptied on read access.			
7:0	RXDATAP0	0x0	R	**RX Data 0 Peek**
	First byte read from buffer. Buffer is not emptied on read access.			

23.5.14 I2C_TXDATA - Transmit Buffer Data Register

Offset	Bit Position																													
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									TXDATA							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	TXDATA	0x0	W	**TX Data**
	Use this register to write a byte to the transmit buffer.			

23.5.15 I2C_TXDOUBLE - Transmit Buffer Double Data Register

Offset	Bit Position																													
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0								0x0							
Access																	W								W							
Name																	TXDATA1								TXDATA0							

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	TXDATA1	0x0	W	**TX Data**
	Second byte to write to buffer.			
7:0	TXDATA0	0x0	W	**TX Data**
	First byte to write to buffer.			

23.5.16 I2C_IF - Interrupt Flag Register

Offset	Bit Position																															
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access												RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name												SDAERR	SCLERR	CLERR	RXFULL	SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK	RXDATAV	TXBL	TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	SDAERR	0x0	RW	**SDA Error Interrupt Flag**
	Set when the SDA at loopback path is not equal to SDA output			
19	SCLERR	0x0	RW	**SCL Error Interrupt Flag**
	Set when the SCL at loopback path is not equal to SCL output			
18	CLERR	0x0	RW	**Clock Low Error Interrupt Flag**
	Set when the clock is pulled low before a START or a STOP condition could be transmitted.			
17	RXFULL	0x0	RW	**Receive Buffer Full Interrupt Flag**
	Set when the receive buffer becomes full.			
16	SSTOP	0x0	RW	**Slave STOP condition Interrupt Flag**
	Set when a STOP condition has been received. Will be set regardless of the slave being involved in the transaction or not.			
15	CLTO	0x0	RW	**Clock Low Timeout Interrupt Flag**
	Set on each clock low timeout. The timeout value can be set in CLTO bit field in the I2Cn_CTRL register.			
14	BITO	0x0	RW	**Bus Idle Timeout Interrupt Flag**
	Set on each bus idle timeout. The timeout value can be set in the BITO bit field in the I2Cn_CTRL register.			
13	RXUF	0x0	RW	**Receive Buffer Underflow Interrupt Flag**
	Set when data is read from the receive buffer through the I2Cn_RXDATA register while the receive buffer is empty. It is also set when data is read through the I2Cn_RXDOUBLE while the buffer is not full.			
12	TXOF	0x0	RW	**Transmit Buffer Overflow Interrupt Flag**
	Set when data is written to the transmit buffer while the transmit buffer is full.			
11	BUSHOLD	0x0	RW	**Bus Held Interrupt Flag**
	Set when the bus becomes held by the I2C module.			
10	BUSERR	0x0	RW	**Bus Error Interrupt Flag**
	Set when a bus error is detected. The bus error is resolved automatically, but the current transfer is aborted.			
9	ARBLOST	0x0	RW	**Arbitration Lost Interrupt Flag**
	Set when arbitration is lost.			
8	MSTOP	0x0	RW	**Master STOP Condition Interrupt Flag**

Bit	Name	Reset	Access	Description
	Set when a STOP condition has been successfully transmitted. If arbitration is lost during the transmission of the STOP condition, then the MSTOP interrupt flag is not set.			
7	NACK	0x0	RW	**Not Acknowledge Received Interrupt Flag**
	Set when a NACK has been received.			
6	ACK	0x0	RW	**Acknowledge Received Interrupt Flag**
	Set when an ACK has been received.			
5	RXDATAV	0x0	RW	**Receive Data Valid Interrupt Flag**
	Set when received data is half full			
4	TXBL	0x0	RW	**Transmit Buffer Level Interrupt Flag**
	if TXBIL==0, set when the transmit buffer is empty. if TXBIL==1, set when the transmit is half full			
3	TXC	0x0	RW	**Transfer Completed Interrupt Flag**
	Set when the transmit shift register becomes empty and there is no more data in the transmit buffer.			
2	ADDR	0x0	RW	**Address Interrupt Flag**
	Set when incoming address is accepted, i.e. own address or general call address is received.			
1	RSTART	0x0	RW	**Repeated START condition Interrupt Flag**
	Set when a repeated start condition is detected.			
0	START	0x0	RW	**START condition Interrupt Flag**
	Set when a start condition is successfully transmitted.			

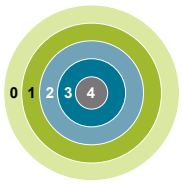
23.5.17 I2C_IEN - Interrupt Enable Register

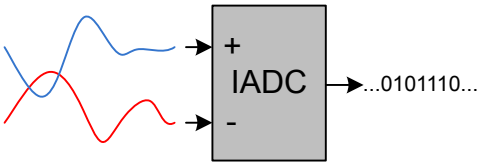
Offset	Bit Position																															
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access												RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name												SDAERR	SCLERR	CLERR	RXFULL	SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK	RXDATAV	TXBL	TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	SDAERR	0x0	RW	**SDA Error Interrupt Flag**
	Set when SDA at loopback path is not equal to SDA output			
19	SCLERR	0x0	RW	**SCL Error Interrupt Flag**
	Set when SCL at loopback path is not equal to SCL output			
18	CLERR	0x0	RW	**Clock Low Error Interrupt Flag**
	Set when the clock is pulled low before a START or a STOP condition could be transmitted.			
17	RXFULL	0x0	RW	**Receive Buffer Full Interrupt Flag**
	Set when the receive buffer becomes full.			
16	SSTOP	0x0	RW	**Slave STOP condition Interrupt Flag**
	Set when a STOP condition has been received. Will be set regardless of the slave being involved in the transaction or not.			
15	CLTO	0x0	RW	**Clock Low Timeout Interrupt Flag**
	Set on each clock low timeout. The timeout value can be set in CLTO bit field in the I2Cn_CTRL register.			
14	BITO	0x0	RW	**Bus Idle Timeout Interrupt Flag**
	Set on each bus idle timeout. The timeout value can be set in the BITO bit field in the I2Cn_CTRL register.			
13	RXUF	0x0	RW	**Receive Buffer Underflow Interrupt Flag**
	Set when data is read from the receive buffer through the I2Cn_RXDATA register while the receive buffer is empty. It is also set when data is read through the I2Cn_RXDOUBLE while the buffer is not full.			
12	TXOF	0x0	RW	**Transmit Buffer Overflow Interrupt Flag**
	Set when data is written to the transmit buffer while the transmit buffer is full.			
11	BUSHOLD	0x0	RW	**Bus Held Interrupt Flag**
	Set when the bus becomes held by the I2C module.			
10	BUSERR	0x0	RW	**Bus Error Interrupt Flag**
	Set when a bus error is detected. The bus error is resolved automatically, but the current transfer is aborted.			
9	ARBLOST	0x0	RW	**Arbitration Lost Interrupt Flag**
	Set when arbitration is lost.			
8	MSTOP	0x0	RW	**Master STOP Condition Interrupt Flag**

Bit	Name	Reset	Access	Description
	Set when a STOP condition has been successfully transmitted. If arbitration is lost during the transmission of the STOP condition, then the MSTOP interrupt flag is not set.			
7	NACK	0x0	RW	**Not Acknowledge Received Interrupt Flag**
	Set when a NACK has been received.			
6	ACK	0x0	RW	**Acknowledge Received Interrupt Flag**
	Set when an ACK has been received.			
5	RXDATAV	0x0	RW	**Receive Data Valid Interrupt Flag**
	Set when data is available in the receive buffer. Cleared automatically when the receive buffer is read.			
4	TXBL	0x0	RW	**Transmit Buffer Level Interrupt Flag**
	Set when the transmit buffer becomes empty. Cleared automatically when new data is written to the transmit buffer.			
3	TXC	0x0	RW	**Transfer Completed Interrupt Flag**
	Set when the transmit shift register becomes empty and there is no more data in the transmit buffer.			
2	ADDR	0x0	RW	**Address Interrupt Flag**
	Set when incoming address is accepted, i.e. own address or general call address is received.			
1	RSTART	0x0	RW	**Repeated START condition Interrupt Flag**
	Set when a repeated start condition is detected.			
0	START	0x0	RW	**START condition Interrupt Flag**
	Set when a start condition is successfully transmitted.			

24. IADC - Incremental Analog to Digital Converter





Quick Facts

What?

The IADC is used to convert analog voltages into a digital representation and features high-speed, low-power operation.

Why?

In many applications there is a need to measure analog signals and record them in a digital representation, without exhausting the energy source.

How?

The low power IADC samples one or more input channels in a programmable sequence. With the help of PRS and DMA, the IADC can operate without CPU intervention in EM2 and EM3, minimizing the number of powered up resources. The IADC can be automatically shut down between conversions to further reduce the energy consumption.

24.1 Introduction

The IADC uses an Incremental Analog to Digital architecture, with a resolution of 12 bits when operating at one million samples per second (1 Msps). The flexible incremental architecture uses oversampling to allow applications to trade speed for higher resolution. An integrated input multiplexer can select from external I/Os and several internal signals.

24.2 Features

- Flexible oversampled architecture allows for tradeoffs between speed and resolution.
  - 1 Msps with oversampling ratio = 2
  - 555 ksps with oversampling ratio = 4
- Digital post-averaging
- Internal and external conversion trigger sources
  - Immediate (software triggered)
  - Local IADC timer
  - External TIMER module (synchronous with output / PWM generation)
  - General PRS hardware signal
- Integrated prescaler for conversion clock generation
- Can be run during EM2 and EM3, waking up the system on interrupts as needed
- Selectable reference sources
  - 1.21 V internal reference
  - External precision reference
  - Analog supply
- Support for offset and gain calibration
- Programmable input gain: 0.5x, 1x, 2x, 3x, or 4x
- Flexible output formatting
  - Unipolar or 2's complement bipolar data
  - Results can be saved in 12 bit, 16 bit, or 20 bit format
  - Programmable left or right justification
  - Optional channel ID tag
- Digital window comparison function detects when results are inside/outside a programmable window
- Two independent groups of configuration registers for setting IADC mode, clock prescaler, reference selection, oversample rate, unipolar/bipolar output formatting, and analog gain
- Programmable single channel conversion
  - Can use either configuration group
  - Triggered by any conversion trigger source
  - Can be tailgated after a scan sequence
  - One shot or continuous mode
  - Local 4-entry FIFO for immediate data storage
  - Programmable watermark level to generate interrupt or initiate DMA transfer
  - Supports overflow and underflow interrupt generation
  - Supports window compare function
- Autonomous multi-channel scan
  - Up to 16 configurable slots in scan sequence
  - Each slot allows independent selection of configuration group, channel selection, and window compare enable
  - Triggered by any conversion trigger source
  - One shot or continuous mode
  - Local 4-entry FIFO for immediate data storage
  - Programmable watermark level to generate interrupt or initiate DMA transfer
  - Supports overflow and underflow interrupt generation
  - Conversion tailgating support for predictable periodic scans

- Available interrupt sources:
  - Single FIFO has DVL (data valid level) entries available (also generates DMA request)
  - Scan FIFO has DVL (data valid level) entries available (also generates DMA request)
  - Single FIFO result compared true for digital compare window
  - Scan FIFO result compared true for digital compare window
  - Single queue conversion has completed
  - Scan queue entry conversion has completed
  - Scan queue table conversion has completed
  - Single FIFO overflow or underflow
  - Scan FIFO overflow or underflow
  - Polarity Error interrupt
  - Port Allocation Error interrupt
  - EM23 clock configuration error

24.3 Functional Description

The incremental ADC module block diagram is shown in Figure 24.1 IADC Overview on page 722.

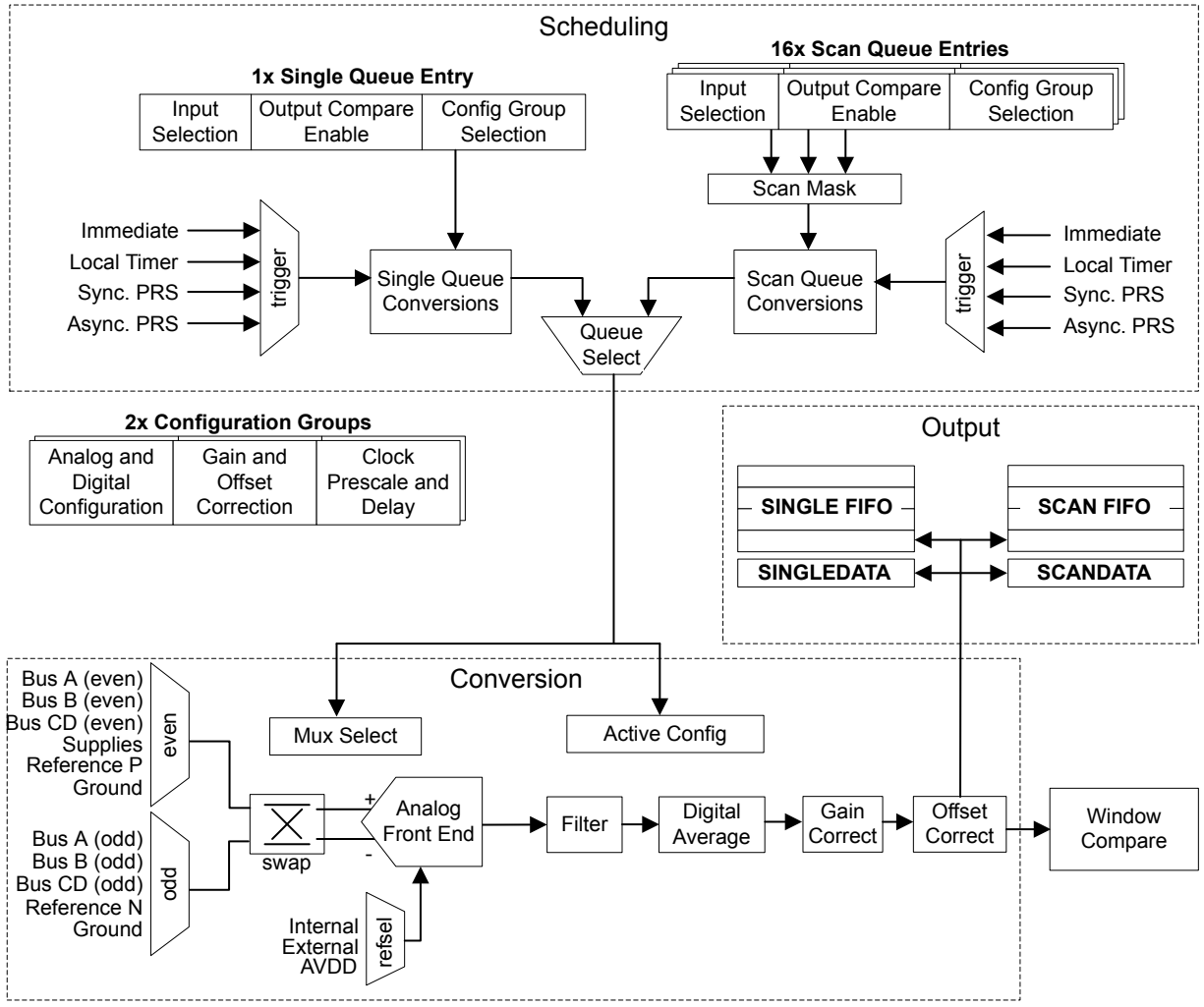


Figure 24.1. IADC Overview

24.3.1 Register Access

Many of the IADC module's configuration registers can only be written while the module is disabled (IADC_EN_EN = 0). These are IADC_CTRL, IADC_TIMER, IADC_CMPTHR, IADC_TRIGGER, IADC_CFGx, IADC_SCALEx, IADC_SCHEDx, and IADC_SCANx. A typical setup sequence for the IADC module is:

1. With the IADC disabled (IADC_EN_EN = 0), program all configuration registers listed above.
2. Enable the IADC by setting EN in IADC_EN to 1.
3. Program the remaining configuration registers.
4. Enable the single or scan queue.
5. The IADC is ready for use.

24.3.2 Clocking

The IADC logic is partitioned into two clock domains: CLK_BUS (APBIF) and CLK_SRC_ADC (CORE). The APBIF domain contains the IADC registers and FIFO read logic. The rest of the IADC is clocked mainly by CLK_SRC_ADC and ADC_CLK, both of which are derived from CLK_CMU_ADC, as shown in Figure 24.2 Clocking on page 724.

CLK_CMU_ADC is the incoming clock routed to the ADC by the CMU, and may be up to 80 MHz. It is selected within the CMU module. If the ADC is to be used synchronously with an external TIMER module, the clock should be configured to derive from the group A clock. If configuring for operation in EM2 or EM3, a clock source available in EM2 and EM3 must be used directly, as the group A clock multiplexer will be shut down in EM2 and EM3.

CLK_SRC_ADC is derived from CLK_CMU_ADC, and must be no faster than 40 MHz. The HSCLKRATE field in IADC_CTRL sets the prescaler to divide CLK_CMU_ADC. If CLK_CMU_ADC is already 40 MHz or slower, HSCLKRATE can be set to 0x0 to pass the clock through to CLK_SRC_ADC without dividing it. CLK_SRC_ADC is the clock source used for the TIMEBASE prescaler as well as the local IADC timer.

ADC_CLK is used to drive the ADC front-end and state machine logic. Another prescaler is used to reduce CLK_SRC_ADC to a suitable frequency for the ADC operating mode. Because the operational mode may be different for single vs. scan conversions, or even for different conversions within a scan, each configuration group has a PRESCALE bit field in the IADC_SCHEDx register. PRESCALE must be set to limit ADC_CLK to no faster than 10 MHz in normal mode for 0.5x and 1x analog gain settings. For analog gain of 2x, 3x, and 4x, the maximum ADC_CLK is 5 MHz, 3.3 MHz, or 2.5 MHz respectively.

Note: If HSCLKRATE is configured to divide CLK_CMU_ADC by more than 1 (HSCLKRATE != 0), then PRESCALE must not be set to divide by 1 (PRESCALE = 0). When this condition is detected, a PRESCALE value of 1 (divide by 2) will be automatically be used instead of the programmed PRESCALE value.

The suspend mode fields IADC_CTRL_ADCCLKSUSPEND0 (for scan conversions) or IADC_CTRL_ADCCLKSUSPEND1 (for single conversions) can be used to shut down the clock between conversions and save power. The ADC logic will wake up the clock before starting IADC warmup and performing a conversion. If the suspend mode is set, the clock will shut down again once the conversion is complete.

When IADC_TRIGGER_SCANTRIGSEL or IADC_TRIGGER_SINGLETRIGSEL is set to IMMEDIATE, IADC_CTRL_ADCCLKSUSPENDn will force the clock to only be running when one of the queues is enabled.

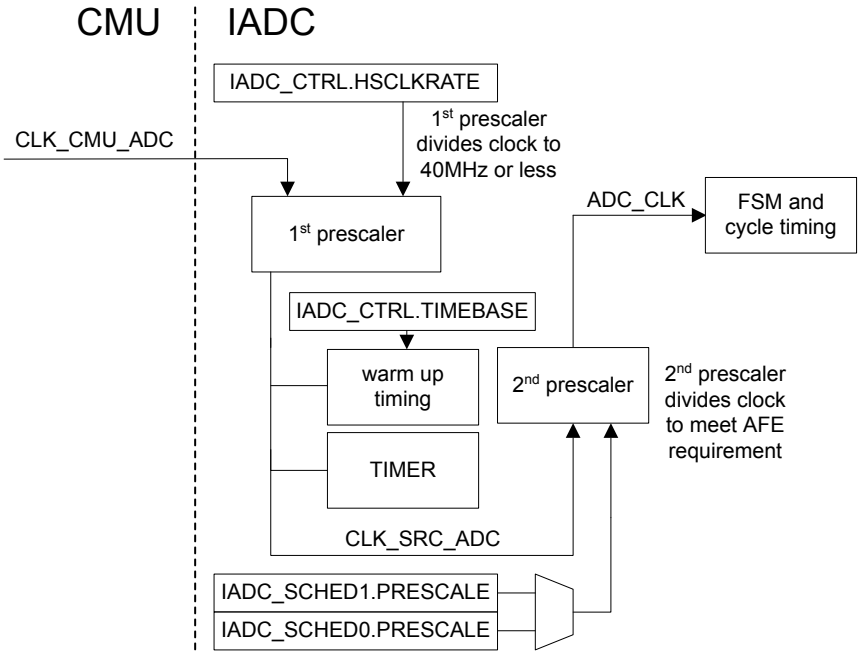


Figure 24.2. Clocking

24.3.3 Conversion Timing

The IADC takes multiple samples of the analog signal to produce each output. The number of input samples contributing to an output word is determined by the oversampling ratio (OSR). Higher OSR settings will improve the ADC's INL and DNL, and reduce system-level noise, but require more time for each conversion. The OSR is configured with the OSRHS bit field in the IADC_CFGx register. Different OSRs may be specified for each configuration group. It is important to note that oversampling is an analog process (pre-digital filter).

24.3.3.1 Warmup Time

To save energy, the IADC can be configured to power down completely or enter a standby state between conversions, if full speed operation is not required for the application. The required ADC warm up time from a full powered-down state is 5 us. Warmup from a standby state requires 1 us. Warmup is automatically timed by the ADC logic when it is required, but software must configure the TIMEBASE field in IADC_CTRL for a minimum 1 us interval. Note that the TIMEBASE counter receives CLK_SRC_ADC, and should be programmed based on that frequency. For example, if CLK_SRC_ADC is 40 MHz, TIMEBASE should be set to at least 0x27 (39) to produce the minimum 1 us interval. When transitioning from a powered-down state, the IADC will use five TIMEBASE intervals. When in standby the IADC will use one TIMEBASE interval.

The WARMUPMODE field in the IADC_CTRL register defines whether the IADC is powered down between conversions (WARMUPMODE = NORMAL), in standby between conversions (WARMUPMODE = KEEPINSTANDBY), or remains powered up (WARMUPMODE = KEEPWARM). The resulting start-up time is shown in Figure 24.3 Start-up Timing on page 725. Note that even in WARMUPMODE = KEEPWARM or KEEPINSTANDBY, the ADC will implement 5 TIMEBASE intervals of warmup on initial power up, or any configuration change affecting PRESCALE, ADCMODE, or REFSEL. IADC_STATUS_ADCWARM reflects the current warmup status of the IADC.

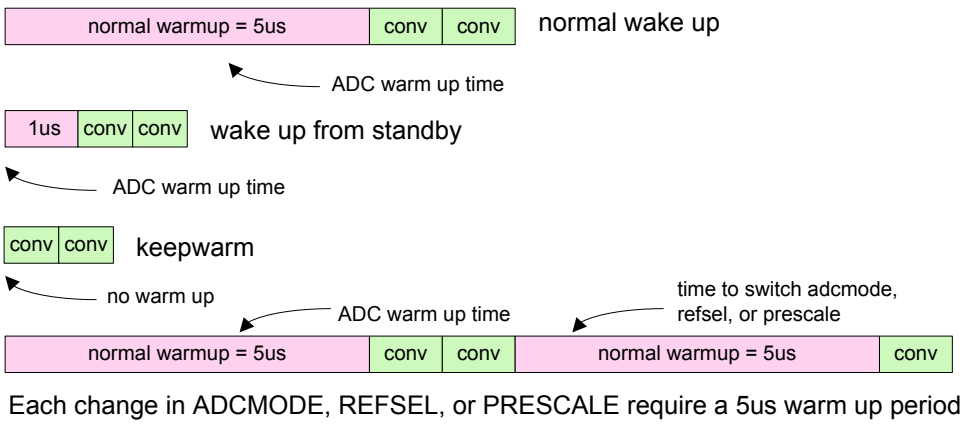


Figure 24.3. Start-up Timing

24.3.3.2 Conversion Pipeline

The IADC uses a pipelined architecture to perform different stages of the ADC conversion in parallel.

The conversion time for a single sample can be determined from the OSR and the pre-scaled CLK_ADC frequency (f_{CLK_ADC}) as:

Conversion Time = ((4 * OSR) + 2) / f_{CLK_ADC}

The minimum OSR is 2, meaning that the fastest possible conversion lasts 10 CLK_ADC clock cycles.

The IADC will automatically insert 2 additional cycles in the pipeline when changing channels to a new GPIO. This allows for hold timing on the previous conversion and allows for time to tristate the ABUS analog buses before connecting the next GPIO to the analog bus. Therefore the maximum sampling rate while continuously sampling on one channel (with CLK_ADC = 10 MHz) is 1 Msps, and the maximum sampling rate while switching channels is 833 ksps. Figure 24.4 Normal ADC Mode Pipeline on page 726 shows both single-channel and channel-switching scenarios powering up from a shutdown state with WARMUPMODE = NORMAL. The 5 us warmup is shown in pink, a first conversion pipeline in green, and a second conversion in orange. The blue area in the top diagram represents the extra time to tristate while changing channels.

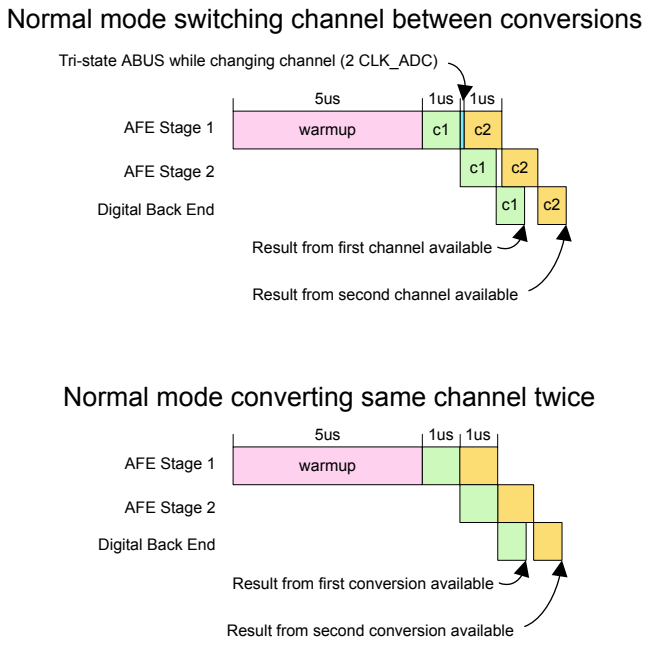


Figure 24.4. Normal ADC Mode Pipeline

24.3.3.3 Scheduling and Triggers

The IADC has several triggering options available for both the Single queue and the Scan queue. When a conversion trigger occurs and there are no other conversions active or pending, the request is serviced immediately. If both the single and scan queues are being used in an application, it is possible to serve the conversion requests as needed, and specify their priority.

Conversion triggering is configured using bit fields in the IADC_TRIGGER register. The SINGLETRIGSEL and SCANTRIGSEL fields specify the trigger source for Single and Scan conversion queues, respectively. The options for trigger source are:

- IMMEDIATE - Trigger from software. This is useful for triggering conversions on-demand from software with no specific sampling frequency requirements, or initiating continuous conversions at full speed.
- TIMER - Use the IADC local timer to trigger conversions. This is useful for triggering conversions at precise intervals.
- PRSCLKGRP - Use a synchronous PRS channel to trigger from an external peripheral in the same clock group domain (i.e. clock group A). This is useful for synchronizing conversions precisely with external TIMER events or PWM outputs.

  Note: It is recommended to configure the PRS consumer registers prior to enabling synchronous PRS triggers to avoid false triggers.

- PRSPOS - Use a positive edge of an asynchronous PRS channel to trigger conversions. The trigger source will require 1-2 ADC_SRC_CLK cycles to synchronize. This is useful for triggering conversions as needed from asynchronous peripheral sources such as GPIO inputs, RTCC events, etc.
- PRSNEG - Use a negative edge of an asynchronous PRS channel to trigger conversions. This is the same as PRSPOS, but operates on negative edges of the selected input.

Both the single and scan trigger sources can be configured to generate one request per trigger, or begin continuous conversions. Setting SINGLETRIGACTION to ONCE will make one conversion request each time the selected single trigger occurs, and a single ADC output will be converted. Setting SINGLETRIGACTION to CONTINUOUS allows the single trigger to begin the first conversion, and when a conversion completes a new one will be requested immediately without requiring a new trigger. Channel selections and configuration should not be changed while SINGLETRIGACTION is set to CONTINUOUS. Doing so can produce conversion errors. The scan queue should be used if channel or configuration switching is required.

The SCANTRIGACTION field works to request conversion scans in a similar manner. Setting SCANTRIGACTION to ONCE will make one request each time the selected scan trigger occurs, and the IADC will perform all conversions specified in the scan once before stopping. Setting SCANTRIGACTION to CONTINUOUS allows the scan trigger to initiate continuous scans. When a scan cycle completes, a new one will be requested immediately without requiring a new trigger.

Conversion priority can be adjusted using the SINGLETAILGATE bit. By default, SINGLETAILGATE is set to TAILGATEOFF, meaning that conversion triggers are queued in the order they are received. Any conversion trigger for the Single queue or the Scan queue will initiate a conversion as soon as possible. If any conversion is already in progress or pending, the new conversion will be handled after the current operation.

Setting SINGLETAILGATE to TAILGATEON gives ultimate priority to the Scan queue. The IADC will only perform single conversions immediately after completion of a scan. This allows systems to use the scan queue for high-priority conversions with tight timing requirements, and the single queue for low-priority, on-demand conversion events. Note that this setting should only be used when scan conversions are guaranteed to trigger. If no scan sequence is triggered, any single conversion trigger will remain pending indefinitely. It is also important to note that if there is not enough time between scan conversions to service a single conversion, the next scan conversion will be delayed.

24.3.3.3.1 Conversion Triggering Examples

Scheduling a Single Sample

The simplest use case for the IADC is performing one conversion on-demand from the Single queue. Figure 24.5 Immediate Single Conversion on page 728 shows the configuration and timing of this use case. The IADC warmup mode is configured for normal (shuts down between conversions). The single queue trigger is configured for immediate triggering of one conversion, and tailgating is turned off. When the conversion is requested (by setting IADC_CMD_SINGLESTART), the IADC block warms up and then begins converting. During the conversion, the CONVERTING bit in IADC_STATUS is set. When the conversion is complete, the queue is disabled, and SINGLEQEN returns low.

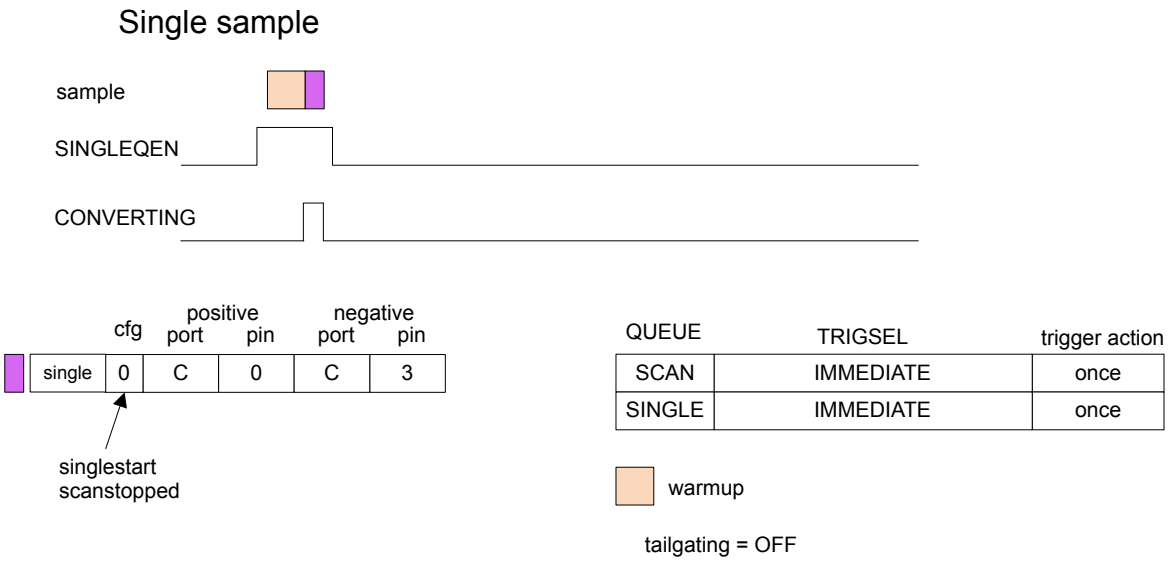


Figure 24.5. Immediate Single Conversion

Periodic Scans

Another common use case is to periodically trigger the IADC to perform a multi-channel scan. Figure 24.6 Periodic Scan Example on page 729 shows the timing of a periodic scan triggered by the IADC's local timer. The scanner is configured to sample four different channels; two using configuration 0 and two using configuration 1. Note that a single TIMER trigger is used to initiate each scan, and all four samples are taken for each trigger. Note also that the IADC inserts another warmup time between conversions 1 and 2, when it switches from configuration 0 to configuration 1. The single queue is disabled and not used in this example.

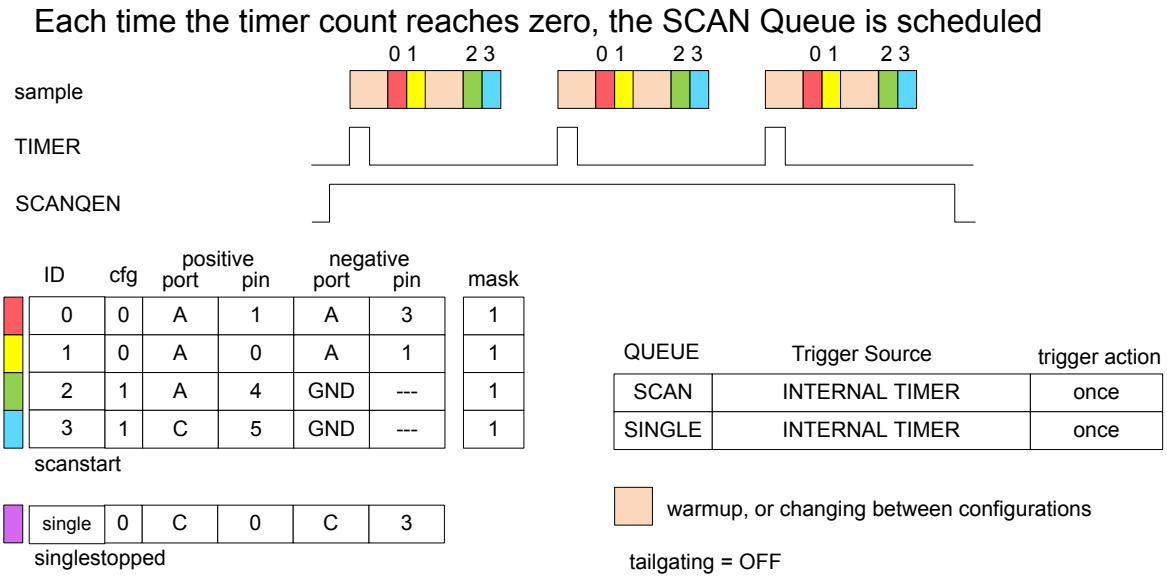


Figure 24.6. Periodic Scan Example

Tailgating Examples

An example using conversion tailgating is shown in Figure 24.7 Simple Conversion with Tailgating Enabled on page 730. In the example, the Scan queue is configured to trigger a two-channel conversion periodically on the IADC local timer, while the Single queue is configured to trigger on-demand from software. When a single conversion is requested, it waits until after the scan sequence is complete, and then the single conversion is performed. The scan conversions are using configuration 0, and the single conversion is using configuration 1, so a warmup delay is inserted between the end of the scan and the beginning of the single conversion cycle. Note that this example provides plenty of time between IADC scan conversions for the single conversion to occur, and no scan conversions are delayed.

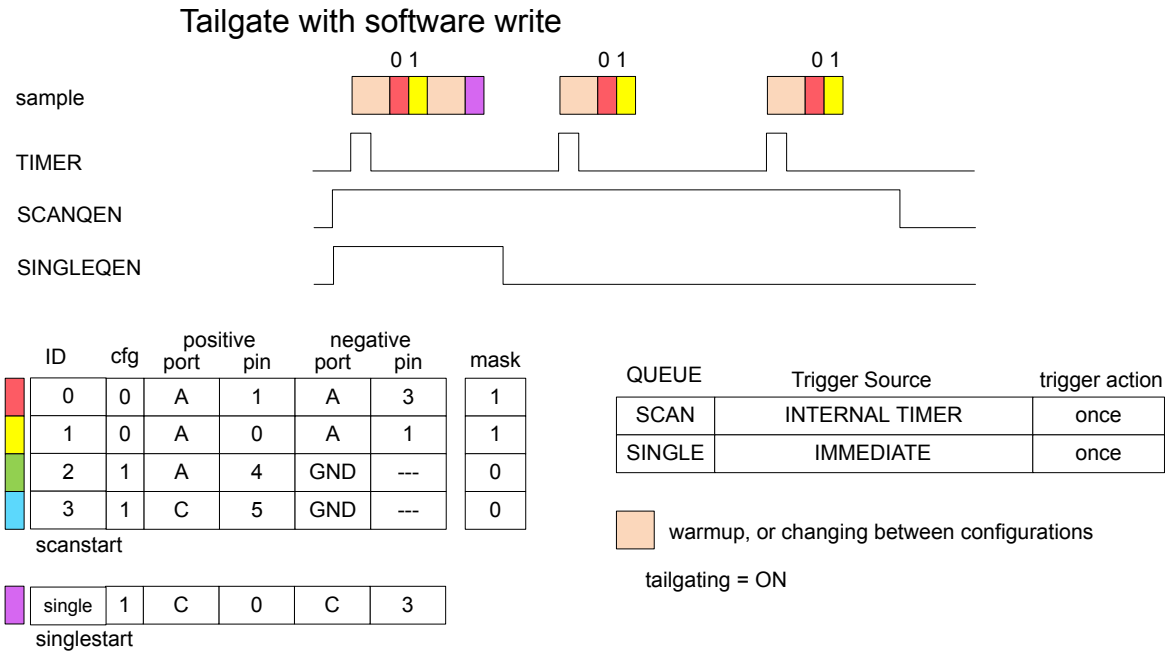


Figure 24.7. Simple Conversion with Tailgating Enabled

Another example, shown in Figure 24.8 Conversions with Tailgating Disabled on page 731, demonstrates how requests are handled on the different conversion queues with tailgating disabled.

In this example, the scan queue is being triggered on the internal timer while the single queue is being triggered on a PRS positive edge. Since tailgating is not enabled, the queues will be serviced on a first come first served basis. The first single queue trigger falls between two scan queue triggers and does not interfere with scan queue timing. The second single queue trigger happens just before the scan queue trigger. The IADC will complete this single queue conversion and delay the next scan queue conversions.

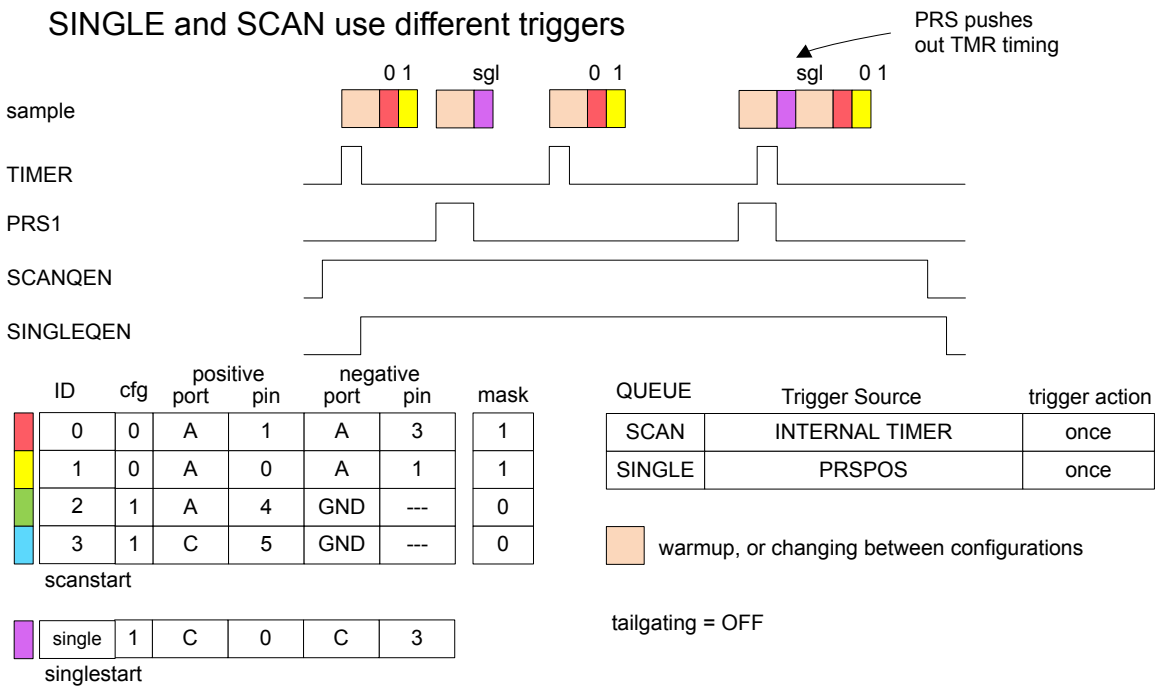


Figure 24.8. Conversions with Tailgating Disabled

Continuous Conversions

An example of continuous conversions triggered from the scan queue is shown in Figure 24.9 Continuous Conversions on page 731. In this example the SCANTRIGACTION field in IADC_TRIGGER is set to CONTINUOUS, and the conversion trigger source is software (SCANTRIGSEL = IMMEDIATE). When the scan queue is enabled with IADC_CMD_SCANSTART, the ADC warms up and then performs repeated back-to-back scans until software disables the scan queue using IADC_CMD_SCANSTOP. While this example shows only one channel converted continuously, it is possible to enable multiple channels for the scan sequence.

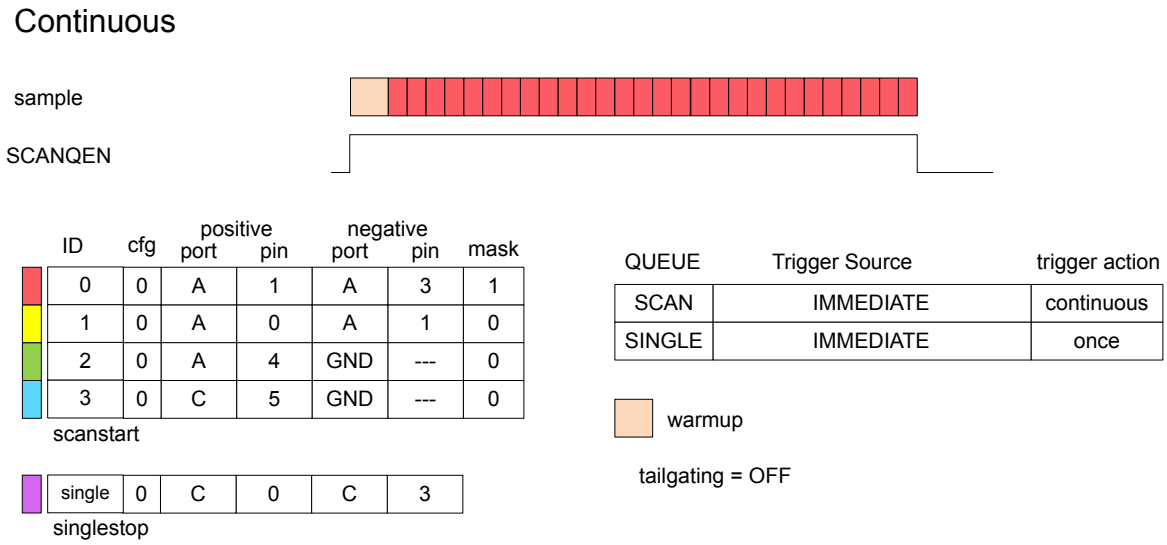


Figure 24.9. Continuous Conversions

24.3.4 Reference Selection and Analog Gain

The default IADC reference is to use the internal band gap circuit. The analog power supply voltage can also be used as a voltage reference. The reference voltage is selected using the REFSEL field in IADC_CFGx. Refer to Table 24.1 Mode Settings on page 732.

Table 24.1. Mode Settings

Reference	Description	Voltage
VBGR	Internal	1.21V
VDDX	Analog Power Supply	AVDD
VREF	External	1.0V - AVDD (1.25V Nominal)

The IADC also has analog gain selection, controlled via the ANALOGGAIN field in IADC_CFGx. The analog gain can be set to 0.5x, 1x, 2x, 3x, or 4x. Note that 2x, 3x, and 4x gain modes require slower ADC_CLK as detailed in 24.3.2 Clocking. The analog gain impacts where the full-scale input reading occurs. For example, with a 1.25 V external reference and ANALOGGAIN set to 2x, the analog input to the IADC is multiplied by a factor of 2, and a full-scale reading occurs at 1.25 V / 2 = 0.625 V. If ANALOGGAIN is set to 0.5x, the full-scale reading of the ADC will not occur until the input reaches 2.5 V. Note that the ADC is only capable of measuring inputs within the supply rails of the device. If the full scale is configured to be greater than the supply voltage, the maximum input will be limited to the supply.

24.3.5 Input and Configuration Selection

The IADC supports measurement on a number of internal and external signals. External signals are routed to GPIO through shared ABUS resources on the device, or (on some devices) through dedicated analog inputs available to the IADC block.

The single queue and the scan queue have separate registers available to select inputs and configurations. The IADC_SINGLE register is used to select the input and configuration for the single queue. The IADC_SCANx registers are used to select the inputs and configurations for each of the scan table entries. In both cases, the register contents and setup are similar. The PORTPOS and PINPOS fields are used to select a signal for the positive ADC input, while PORTNEG and PINNEG are used to select a signal for the negative ADC input. The CFG field selects which of the two configuration sets will be used with the input (i.e. configuration options specified in IADC_CFGx, IADC_SCALEx, and IADC_SCHEDx).

To perform single-ended conversions, the PORTNEG field should be set to GND. This indicates that the positive ADC input will be measured with reference to chip ground. PORTPOS and PINPOS should be used to select the desired input signal. The PINNEG field is not used for single-ended conversions.

To perform differential conversions, PORTPOS, PINPOS are used to select the positive input to the ADC, while PORTNEG and PINNEG are used to select the negative input. Note that there are two independent multiplexers in the ADC, and firmware cannot select two signals from the same multiplexer for a differential measurement. The "even" multiplexer consists of all EVEN ABUS selections, Supply voltage options, GND, and VREFP. The "odd" multiplexer consists of all ODD ABUS selections, GND, and VREFN. One selection from each multiplexer is allowed on the positive and negative input. More detailed examples may be found in 24.3.5.3 Input Selection Examples.

The scan queue has one additional register, IADC_MASKREQ, to specify which of the 16 possible channel slots will be converted during a scan operation. Each channel in the scan queue is enabled by writing the corresponding bit in the IADC_MASKREQ register to 1. Enabled channels will be converted in sequence from lowest to highest, during a scan. See 24.3.5.4 Scan Queue for more details on using the scan queue.

24.3.5.1 External GPIO Inputs

GPIO input selections are routed through shared ABUS resources. In order for the IADC to use any GPIO as an input, the IADC must be allocated appropriate analog bus resources in the GPIO_ABUSALLOC, GPIO_BBUSALLOC, or GPIO_CDBUSALLOC registers. For example, if IADC0 will be using both odd and even numbered pins on GPIO port PA, then AEVEN0 and AODD0 in GPIO_ABUSALLOC could both be set to IADC0. This gives IADC0 access to these two buses. Generally, bus access is set to specific peripherals at configuration time and left alone - it is not normally required to change the bus allocation on the fly. If the IADC requests a pin from a bus that has not been allocated to the IADC, an error will be generated, the PORTALLOCERRIF in IADC_IF will be set, and any conversion result will be 0. For more details on analog bus structure and capabilities, refer to the GPIO section.

When the appropriate analog buses have been configured to route to the IADC, GPIO selection is a simple matter of programming the desired port and pin into the PORTPOS, PINPOS, PORTNEG, and PINNEG fields. For example, to configure a channel to convert the differential voltage between pins PA5 and PA4, PORTPOS = PORTA, PINPOS = 5, PORTNEG = PORTA, PINNEG = 4. If an invalid selection is made, a polarity error will be generated. More specific examples are described in 24.3.5.3 Input Selection Examples.

24.3.5.2 Internal and Dedicated Inputs

Internal signals and dedicated inputs are not routed through the shared ABUS resources. In general, these resources are selected directly by the settings of PORTPOS and PORTNEG, while the PINPOS and PINNEG fields are not used. When PORTPOS is set to SUPPLY, PINPOS is used to select which of the power supplies is connected. To facilitate power supply measurments using internal reference options, all supplies are attenuated by a factor of 4.

Table 24.2. Supply Selection (PORTPOS = SUPPLY)

PINPOS	Supply Connection	Voltage at Positive Input
0	AVDD	AVDD / 4
1	VDDIO	VDDIO / 4
2	VSS	VSS
3	VSS	VSS
4	DVDD	DVDD / 4
7	DECOUPLE	DECOUPLE

24.3.5.3 Input Selection Examples

When configuring to measure a single-ended signal, the positive input selection should always point to the desired input, and PORTNEG should be programmed to GND.

Correct configuration examples for single-ended conversions are shown in Figure 24.10 Single Ended Port/Pin Selection Odd Channel on page 734 and Figure 24.11 Single Ended Port/Pin Selection Even Channel on page 734. Note that the IADC logic will automatically swap the appropriate multiplexer to the positive input of the ADC.

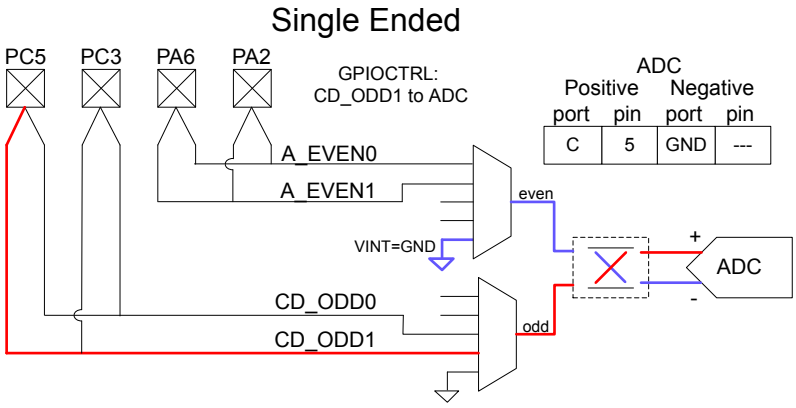


Figure 24.10. Single Ended Port/Pin Selection Odd Channel

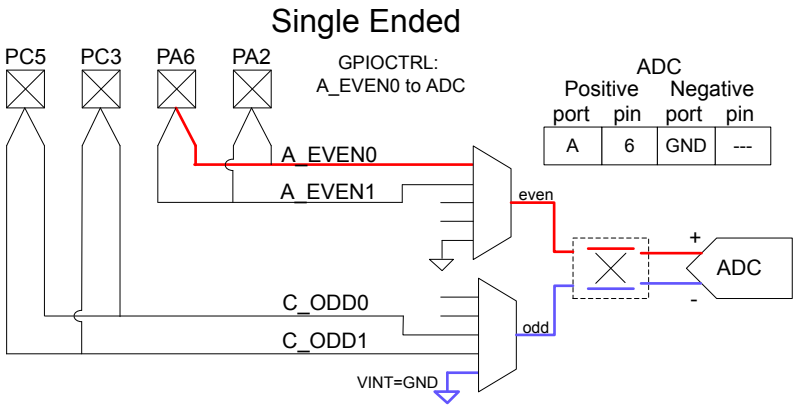


Figure 24.11. Single Ended Port/Pin Selection Even Channel

Figure 24.12 Single Ended Port/Pin Selection Polarity Error on page 735 shows an example where the PORTPOS input has been configured to GND, with PORTNEG and PINNEG configured for a GPIO pin. This will result in a polarity error (POLARITYERRIF in IADC_IF will be set) and any conversion result will be 0.

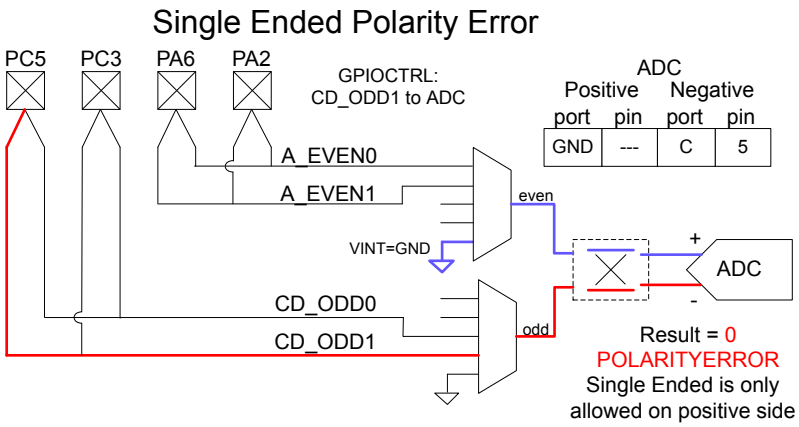


Figure 24.12. Single Ended Port/Pin Selection Polarity Error

Correct configuration examples for differential conversions are shown in Figure 24.13 Differential Port/Pin Selection without Swap on page 735 and Figure 24.14 Differential Port/Pin Selection with Swap on page 736. In both these examples, the inputs were selected from one EVEN multiplexer channel and one ODD multiplexer channel. As with single-ended mode, the IADC logic will automatically swap the multiplexer connections to the IADC input if needed.

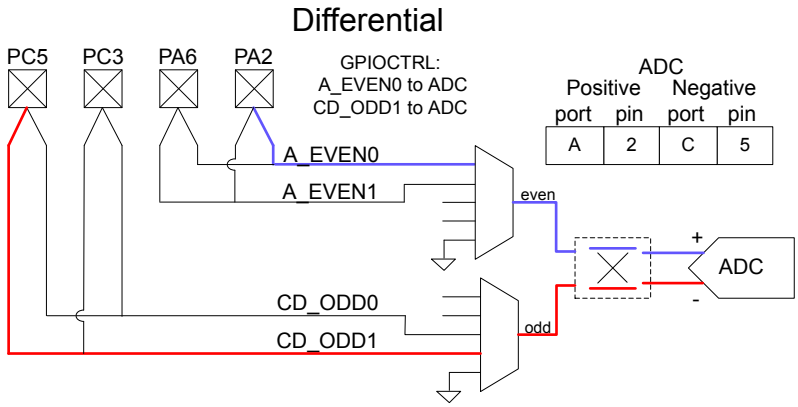


Figure 24.13. Differential Port/Pin Selection without Swap

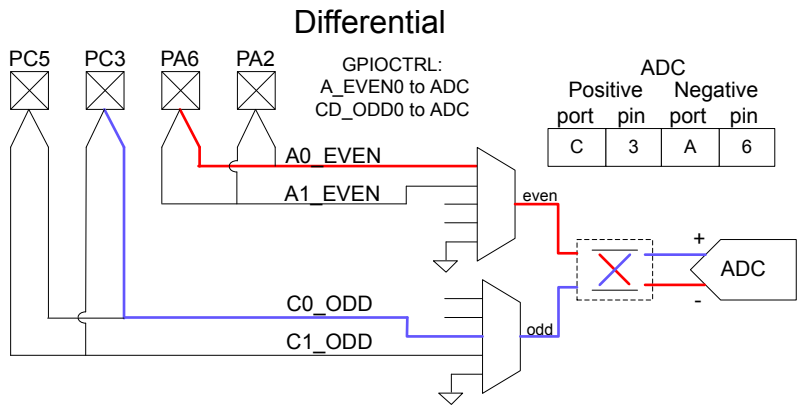


Figure 24.14. Differential Port/Pin Selection with Swap

Figure 24.15 Differential Port/Pin Selection Polarity Error on page 736 shows an example where the both the positive and the negative input selections point to ODD buses. Even though the IADC has been allocated both buses, they both route through the ODD input multiplexer and cannot be measured against one another. This will result in a polarity error (POLARITYERRIF in IADC_IF will be set) and any conversion result will be 0x7FFFF. Likewise, a polarity error will occur if both inputs are selected from EVEN buses.

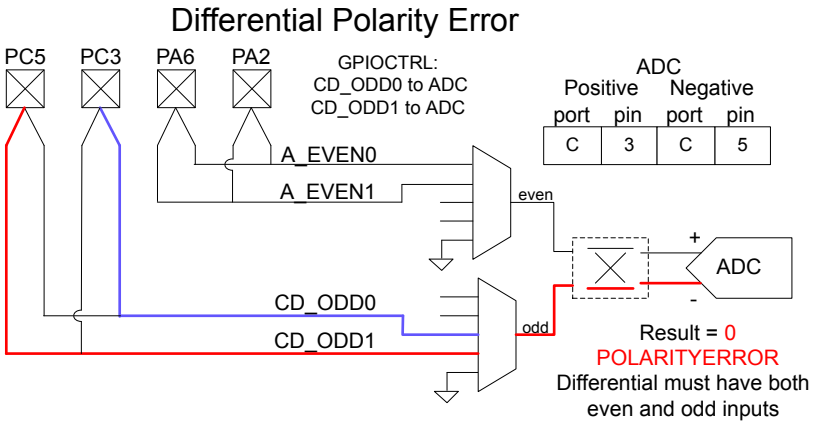


Figure 24.15. Differential Port/Pin Selection Polarity Error

24.3.5.4 Scan Queue

The scan queue allows the IADC to automatically convert up to 16 channels in sequence without CPU intervention. Input and configuration selection for each channel in the scan table is specified by the IADC_SCANx register for that channel (channel 0 is configured with IADC_SCAN0, channel 1 is configured with IADC_SCAN1, and so on). The IADC_MASKREQ register allows software to define which of the scan table entries (IADC_SCANx) to convert during a scan. For example, channels 0, 1, and 7 can be enabled by writing bits 0, 1, and 7 of IADC_MASKREQ to 1 (IADC_MASKREQ = 0x0083).

The IADC_SCANx registers must be configured when the IADC module is disabled (IADC_EN_EN = 0). IADC_MASKREQ can be written while IADC_EN_EN is set to 1. If a scan operation is in progress, MASKREQ will be synchronized and held until the current scan operation has completed. Then MASKREQ is copied into the STMASK register for the next scan operation. IADC_STMASK is the working copy of the MASKREQ used by the IADC during a scan. MASKREQ will only transfer to STMASK when the scan queue is not scanning and converting the scan table. IADC_STATUS_MASKWRITEPENDING can be used by software to see when the MASKREQ write has been transferred to STMASK. Writing a new MASKREQ in the middle of a scan will not corrupt the current scan. Software which writes to MASKREQ during a scan operation must ensure IADC_STATUS_MASKWRITEPENDING returns to 0 before updating IADC_MASKREQ again. Figure 24.16 MASKREQ Updates on page 737 shows a time line of when the MASKREQ write is updated.

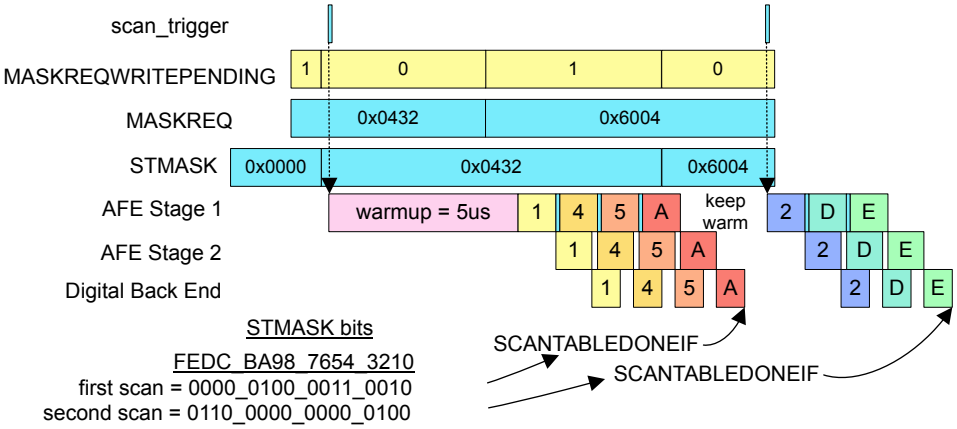


Figure 24.16. MASKREQ Updates

24.3.6 Gain and Offset Correction

The IADC has built in gain and offset correction capabilities. Each of the two configuration groups contains its own correction values stored in the IADC_SCALEx register, allowing the IADC to automatically apply the appropriate correction for the IADC configuration that is being used.

Gain correction is performed through a fixed-point 16-bit value with a range from 0.75x to 1.2499x. The 3 MSBs of the gain value are not directly writeable. The GAIN3MSB bit in IADC_SCALEx is used to select between 011 and 100 for the 3 MSBs, and the lower 13 bits are programmed directly into IADC_SCALEx_GAIN13LSB. Clearing GAIN3MSB to 0 selects the most significant bits of the gain as 011, representing a range from 0.75x to 0.9999x. Setting GAIN3MSB to 1 selects the most significant bits of the gain as 100, representing a range from 1.00x to 1.2499x.

Offset correction is controlled by the OFFSET field in IADC_SCALEx. It is important to note that the offset correction does not have a direct 1-to-1 relationship with the LSB of the IADC output, and depends on both the OSR and gain correction settings. The offset correction range is +/- 12.5% of full scale. OFFSET is encoded as a 2's complement, 18-bit number with the LSB representing $1 / 2^{20}$ of full scale. Thus, bit 8 of OFFSET aligns with bit 0 of the 12-bit IADC output word.

24.3.6.1 Using Production Calibration Parameters

IADC calibration is performed on every device during Silicon Labs production test and production calibration parameters are stored in the flash DI page. The production calibration values are useful for a wide variety of possible IADC configurations, but do not map directly to the offset and gain correction fields in the IADC_SCALEx registers. Software must calculate the actual offset and gain correction values from the factory calibration values.

24.3.6.1.1 Gain Correction

The IADC gain error is designed to be minimal with the digital gain correction set to 1.0 (GAIN3MSB = 1 and GAIN13LSB = 0). Tighter gain error is achieved by adjusting these values in IADC_SCALEx. Using this gain correction mechanism will result in a slight increase to the DNL of the converter, which is reduced by higher OSR settings.

Gain error is measured during production test at various settings of ANALOGGAIN, and stored in the DEVINFO_IADC0GAIN0 and DEVINFO_IADC0GAIN1 locations. The GAINCANA1 field is used for 0.5x and 1x ANALOGGAIN settings, while GAINCANA2, GAINCANA3, and GANCANA4 are used for ANALOGGAIN settings of 2x, 3x, and 4x, respectively. The GAINCANAn values are expressed as the full 16-bit fixed-point gain, and must be compressed before writing to the IADC_SCALEx register.

To apply a factory-calibrated gain:

1. Read the appropriate GAINCANAn field from the DEVINFO locations for the selected ANALOGGAIN.
2. Write the MSB (bit 15) of GAINCANAn to GAIN3MSB in IADC_SCALEx.
3. Write the 13 LSBs (bits 12-0) of GAINCANAn to GAIN13LSB in IADC_SCALEx.

24.3.6.1.2 Offset Correction

Offset is impacted by the selected ANALOGGAIN and OSR settings in IADC_CFGx, the GAIN3MSB and GAIN13LSB values in IADC_SCALEx, and the voltage reference. Offset is production calibrated for any combination of possibilities, but the OFFSET register value must be calculated for the given situation before it can be effectively used. The production offset calibration consists of four 16-bit terms written to the DEVICEINFO space in the IADCOFFSETCAL0 and IADCOFFSETCAL1 locations: OFFSETANA1NORM, OFFSETANA2NORM, OFFSETANA3NORM, and OFFSETANABASE. The following procedures will determine the setting for the OFFSET register based on production calibration values.

Step 1: Determine the offset gain adjustment term (off_gain) based on ANALOGGAIN.

For ANALOGGAIN set to 0.5x or 1x:

off_gain = 0

For ANALOGGAIN set to 2x, 3x, or 4x, off_gain is calculated as:

off_gain = OFFSETANA2NORM * (gain - 1)

This is summarized in Table 24.3 Offset Gain Adjustment on page 739.

Table 24.3. Offset Gain Adjustment

ANALOGGAIN Setting	Analog front-end gain	Offset Gain Adjustment Term (off_gain)
ANAGAIN0P5	0.5 x	0
ANAGAIN1	1 x	0
ANAGAIN2	2 x	OFFSETANA2NORM * 1
ANAGAIN3	3 x	OFFSETANA2NORM * 2
ANAGAIN4	4 x	OFFSETANA2NORM * 3

Step 2: Calculate the analog offset adjustment term (off_ana) based on OSR and off_gain.

For an OSR of 2x (OSRHS = 0):

off_ana = OFFSETANA1NORM + off_gain

For all other OSR settings, 4x - 64x:

off_ana = OFFSETANABASE + 2*(OFFSETANA3NORM - off_gain)/OSR

The following table expresses these equations:

Table 24.4. Analog Offset Adjustment

OSRHS Setting	OSR	Analog Offset Adjustment Term (off_ana)
HISPD2	2 x	OFFSETANA1NORM + off_gain
HISPD4	4 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/2
HISPD8	8 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/4
HISPD16	16 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/8
HISPD32	32 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/16
HISPD64	64 x	OFFSETANABASE + (OFFSETANA3NORM - off_gain)/32

Step 3: Compensate for reference voltage differences.

The off_ana term represents the offset at the input of the ADC, meaning that the reference voltage will have an impact on the magnitude of the offset at the output. Production calibration values are determined with a 1.25 V reference source. If a voltage significantly different than 1.25 V is used for V_{REF}, adjust the off_ana term by a factor of 1.25 / V_{REF}.

off_ana = off_ana * (1.25 / V_{REF})

Step 4: Calculate total offset by adding the analog offset to the systematic offset.

Systematic offset is a fixed number dependent on OSR, and calculated according to the following equation:

off_sys = 640*(256/OSR)

Total uncorrected offset (off_tot) is calculated by:

off_tot = (off_ana * 4 + off_sys)

Step 5: Apply gain error correction, if needed.

Before writing the OFFSET field, the total uncorrected offset must be multiplied by the gain calibration factor. If the gain calibration factor is equal to 1.0 (0x8000 in 16-bit hex, or GAIN3MSB = 1 and GAIN13LSB = 0), this step may be skipped. Otherwise, adjust off_tot according to the following equation:

off_tot = GAIN_FACTOR * (off_tot + 0x80000) - 0x80000

where GAIN_FACTOR = GAINCANAn / 32768.

Step 6: Write the offset correction value to the OFFSET field.

The OFFSET field holds an 18-bit 2's complement number, which should be the negation of the total offset, or -(off_tot). Before writing to the SCALE register, any leading sign bits should be masked off to avoid corrupting the programmed gain settings.

OFFSET = 0x3FFFF & (-off_tot)

24.3.6.2 Calibration

Calibration can be performed in-system to correct for external errors and provide more accurate measurements. The general calibration procedure is as follows:

1. Configure the ADC to the desired mode, OSR, analog gain settings, reference source, etc.
2. Force the IADC to use bipolar output for the conversion: IADC_CFGx_TWOSCOMPL = FORCEBIPOLAR.
3. Set the initial offset to the maximum negative value (IADC_SCALEx_OFFSET = 0x20000), and the initial gain to 1.0 (GAIN3MSB = 1, GAIN13LSB = 0x0000). This will prevent output saturation when measuring full scale.
4. Apply a full-scale positive input to the IADC and perform a conversion (*result_fullscale*). Multiple conversions can be performed and averaged together to reduce any system-level noise.
5. Apply a zero input to the IADC and perform a conversion (*result_zero*). Multiple conversions can be performed and averaged together to reduce any system-level noise.
6. Calculate the gain correction factor: Divide the expected value by the difference in the measured values (*result_fullscale* - *result_zero*). Note that the offset adjustment in Step 3 will be canceled out by this calculation.
7. Write the gain correction factor to the IADC using the GAIN3MSB and GAIN13LSB fields in IADC_SCALEx.
8. Set IADC_SCALEx_OFFSET to 0x00000 in preparation for the offset calibration.
9. Apply the desired zero voltage to the IADC input and perform a conversion (*result_offset*). Multiple conversions can be performed and averaged together to reduce any system-level noise.
10. Multiply *result_offset* to convert to a 20-bit value (*result_offset_20*). For example, a 12-bit result should be multiplied by 256.
11. Negate *result_offset_20* and write the value to IADC_SCALEx_OFFSET.

    Note that the IADC_SCALEx_OFFSET field is 18 bits. If the result is greater than (2^{17} - 1) or less than (-2^{17}), the offset is too large to be corrected.

24.3.7 Output Data FIFOs

The single and scan queues each have a four-word data FIFO. Conversions results are written to the output data FIFO associated with the queue. Single queue results are written to the single FIFO and scan queue results are written to the scan data FIFO. The two queues are identical in operation, but independent.

Conversion results are read from the single FIFO using IADC_SINGLEFIFODATA. Reading SINGLEFIFODATA will pop the oldest result from the FIFO. It is also possible to read the most recent valid data word using IADC_SINGLEDATA. Reading SINGLEDATA does not pop a conversion from the FIFO. Similarly, the scan FIFO results are read with IADC_SCANFIFODATA, which reads the oldest result and pops the FIFO. The most recent scan result can be read using IADC_SCANDATA.

When the single FIFO has valid data, the SINGLEFIFODV flag in IADC_STATUS is set to 1. When the scan FIFO has valid data SCANFIFODV in IADC_STATUS is set to 1. These data valid status bits are cleared automatically whenever the associated FIFO is empty. For more granular FIFO status, the number of data words present in the FIFO is indicated in IADC_SINGLEFIFOSTAT (for single FIFO) or IADC_SCANFIFOSTAT (for scan FIFO).

A programmable data level watermark is also available for the FIFOs, allowing hardware to trigger interrupts or DMA operations when a specified number of conversion results are available. The DVL field in register SINGLEFIFOCFG or SCANFIFOCFG sets the watermark level, between 1 and 4 conversions. If the number of valid entries in the FIFO reaches or exceeds the level set in DVL, the SINGLEFIFODVLIF (for single FIFO) or SCANFIFODVLIF (for scan FIFO) flag in the IADC_IF register will be set to 1. If enabled, an interrupt or DMA request will be triggered when the flag is set.

By default, DMA requests are turned off for operation in EM2 or EM3. However, the DMAWUFIFOSINGLE or DMAWUFIFOSCAN bits in SINGLEFIFOCFG or SCANFIFOCFG may be used to enable DMA operations in these lower energy modes.

Overflow and underflow status flags are also available in IADC_IF. An overflow condition occurs when an IADC conversion completes, but the associated FIFO is already full. In an overflow case the SINGLEFIFOOFIF or SCANFIFOOFIF flag will be set. The most recent conversion will still be available in the SINGLEDATA or SCANDATA register, but the FIFO will not be updated with the new data. An underflow condition occurs when software or hardware attempts to read from an empty FIFO. In an underflow case the SINGLEFIFOUFIF or SCANFIFOUFIF flag will be set.

24.3.7.1 Data Alignment and Channel ID

The IADC has data alignment options and the ability to include a channel ID along with the conversion data. For the single queue, alignment and channel ID are configured in the IADC_SINGLEFIFOCFG register. For the scan queue, alignment and channel ID are configured in the IADC_SCANFIFOCFG register.

The ALIGNMENT bit field specifies the data justification and the number of data bits as shown in Figure 24.17 Data Alignment on page 742. By default, the converter will produce 12-bit right-justified data, corresponding to ALIGNMENT = RIGHT12.

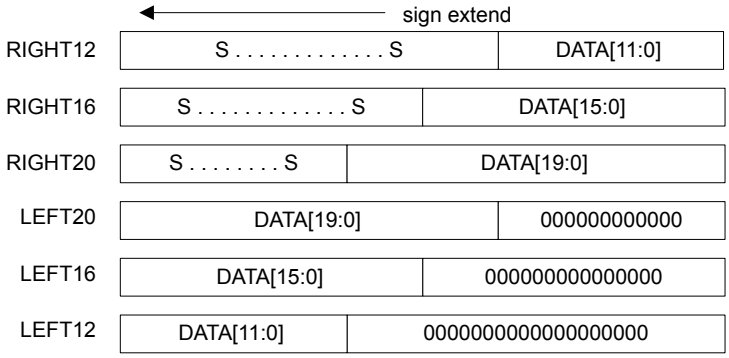


Figure 24.17. Data Alignment

The SHOWID bit controls whether the conversion channel ID is included in the output data word. This option is primarily used with the scan FIFO to help software determine which channel each conversion result came from. If SHOWID is enabled for single conversions, the ID will always be set to 0x20. Figure 24.18 Data Alignment With ID on page 742 shows output data formatting including the ID, when SHOWID = 1.

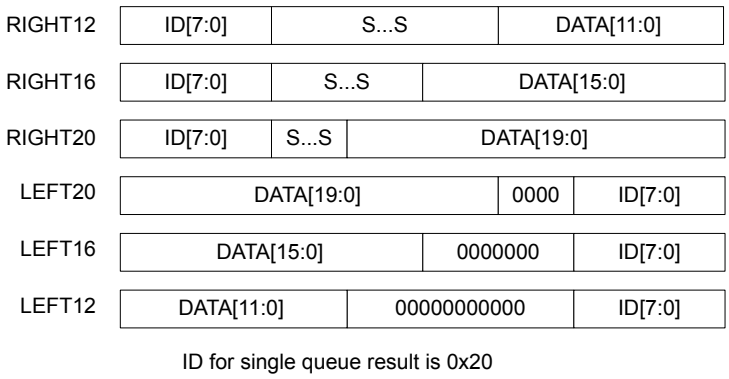


Figure 24.18. Data Alignment With ID

24.3.7.2 Output Polarity

The output polarity of the IADC is controlled by the TWOSCOMPL field in the IADC_CFGx register. The IADC supports unipolar and bipolar output formatting independent of the input configuration. By default, the TWOSCOMPL field is set to AUTO, meaning that single-ended conversions will produce unipolar output, and differential conversions will produce bipolar output. The polarity can be forced to unipolar or bipolar mode by setting TWOSCOMPL to FORCEUNIPOLAR or FORCEBIPOLAR, respectively.

Unipolar samples are unsigned integers representing zero to positive full-scale. Bipolar samples are two's-complement signed integers, representing negative full-scale to positive full-scale. Using unipolar mode on a differential input signal allows for more dynamic range when the signal is positive, but will saturate to zero when the signal is negative.

Note: If bipolar output is used with a single-ended input configuration, it is possible to see negative output values when the input is close to ground. However, the input voltage is still limited by the supply range of the device.

24.3.7.3 Digital Accumulation and Averaging

The IADC may optionally accumulate and average several conversion results before posting an output word to the FIFO. Digital averaging is controlled by the DIGAVG field in the IADC_CFGx register. It can be configured to average 1, 2, 4, 8, or 16 samples. The IADC will collect the number of samples specified by DIGAVG on the selected channel slot back-to-back, and produce only one averaged output word.

24.3.7.4 Output Resolution

The usable output resolution of the IADC is a minimum of 12 bits, when the oversampling ratio is set to 2 and no digital averaging is used (DIGAVG = AVG1). An extra bit of output resolution is produced for every power of 2 increase in either of these settings. In other words, the output resolution of the ADC can be determined as:

$$\text{Output Resolution} = 11 + \log_2(\text{OversamplingRatio} \times \text{DigitalAveraging})$$

The MSB is always left-aligned within the DATA field, and the output word will be truncated to 12, 16, or 20 bits, as shown in Figure 24.17 Data Alignment on page 742 and Figure 24.18 Data Alignment With ID on page 742. When using 16 or 20 bit alignment with lower oversampling ratio and digital averaging settings, LSBs of the output can contain residual effects of the offset and gain computation. These residual effects do not represent additional information about the input signal. Any extra LSBs can be masked to 0 by software.

Table 24.5. Output Resolution Masking Examples

Alignment Setting	Oversampling Ratio	Digital Averaging	Number of averaged samples	Output Resolution	Recommended Mask for DATA field
16-bit	2x	1x	2	12 bits	0xFFF0
16-bit	8x	2x	16	15 bits	0xFFFE
20-bit	2x	1x	2	12 bits	0xFFF00
20-bit	16x	4x	64	17 bits	0xFFFF8

24.3.7.5 Flushing the FIFOs

Each FIFO has a command bit in the IADC_CMD register that can be used to trigger a FIFO flush. The FIFO data may be flushed independently for each queue. To flush a FIFO:

1. The IADC must be enabled with the clock running.
2. Disable the queue associated with the FIFO using the SCANSTOP or SINGLESTOP bits in the IADC_CMD register.
3. Ensure the queue is disabled by reading the associated flag in the IADC_STATUS register (SINGLEQEN or SCANQEN).
4. Set the command bit to flush the desired FIFO (SINGLEFIFOFLUSH or SCANFIFOFLUSH) in the IADC_CMD register.
5. Wait for the corresponding status bit (SINGLEFIFOFLUSHING or SCANFIFOFLUSHING) in IADC_STATUS to go low.

24.3.8 Window Compare

The IADC has a window comparison unit that can trigger interrupts conditional on the output data of the converter. The window comparison unit has two thresholds - greater than or equal (ADGT), and less than or equal (ADLT), which are programmable through the IADC_CMPTHR register. The ADGT and ADLT thresholds always use a 16 bit, left-justified format, regardless of the format specified by the FIFO. The 12-bit conversion result will be compared against the upper 12 bits of the window comparator.

The window comparison unit is active on the ADC output on a conversion-by-conversion basis, and is shared between the two FIFOs. It is not possible to set different window comparison thresholds for different channels or for each FIFO. However, each channel specified in the IADC has a CMP bit field to enable the window comparison on results from that channel. For example, it is possible to only apply the window comparison and associated interrupt to scan channel #3 by setting the CMP field in IADC_SCAN3 to 1. When the CMP field associated with a channel is 0, the window comparator will not be active for results from that channel.

The window comparator supports conditional triggering on output results which are inside or outside a specified window. When ADLT is greater than or equal to ADGT, the comparator will trigger on an "inside" condition, or when DATA <= ADLT **and** DATA >= GT. When ADLT is less than ADGT, the comparator will trigger on an "outside" condition, or when DATA <= ADLT **or** DATA >= GT.

Figure 24.19 Window Comparison Examples on page 744 shows different configurations of the ADLT and ADGT values and the resulting windows. When the window comparator detects that the appropriate conditions are met (shown by the shaded region in the figure), it will generate an interrupt via the SINGLECMPIF flag for conversions on the single queue, or via the SCANCMPIF flag for conversions on the scan queue.

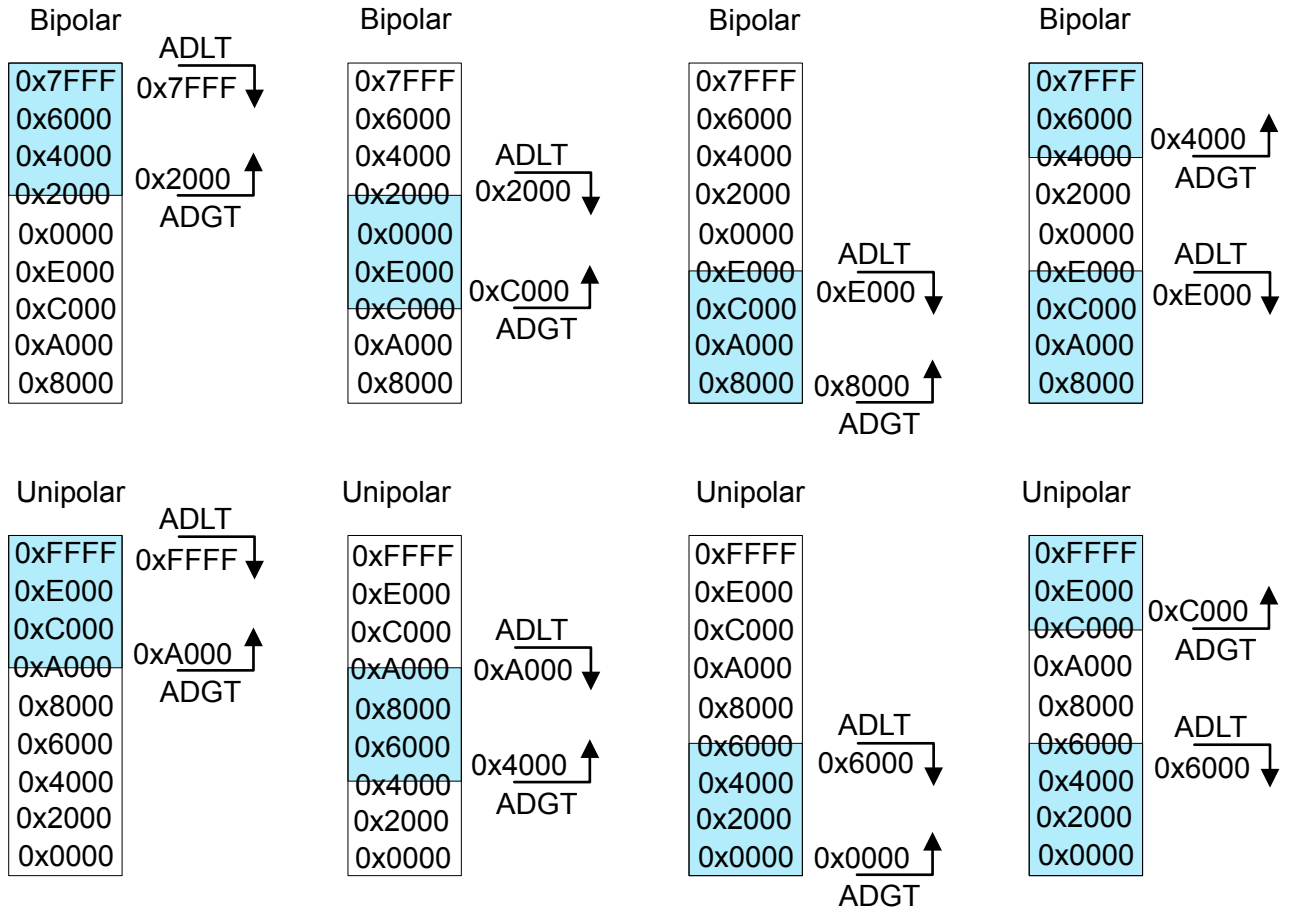


Figure 24.19. Window Comparison Examples

24.3.9 Interrupts

Interrupts are enabled in the IADC_IEN register, allowing interrupts to be generated on several different IADC conditions. Each of the flags in IADC_IF has a corresponding enable bit in the IADC_IEN register. A brief overview of the available interrupt sources is shown in the list below; more details can be found in the relevant sections of this chapter.

- SINGLEFIFODVLIF - The single FIFO watermark specified in SINGLEFIFOCFG_DVL has been reached or exceeded.
- SCANFIFODVLIF - The scan FIFO watermark specified in SCANFIFOCFG_DVL has been reached or exceeded.
- SINGLECMPIF - A conversion result from the single queue tripped the window comparator.
- SCANCMPIF - A conversion result from the scan queue tripped the window comparator.
- SCANENTRYDONEIF - A scan queue conversion has completed.
- SCANTABLEDONEIF - A scan queue operation has completed (all channels specified in the scan mask have been converted once).
- POLARITYERRIF - A channel polarity selection error has occurred (two channels from the EVEN multiplexer or two channels from the ODD multiplexer were selected for positive and negative inputs).
- PORTALLOCERRIF - A port allocation error has occurred (a pin not allocated to the IADC in the GPIO bus allocation registers was requested).
- SINGLEFIFOOFIF - A single FIFO overflow has occurred.
- SCANFIFOOFIF - A scan FIFO overflow has occurred.
- SINGLEFIFOUFIF - A single FIFO underflow has occurred.
- SCANFIFOUFIF - A scan FIFO underflow has occurred.
- EM23ABORTERRORIF - The system entered EM2 or EM3 while the IADC was converting and using a clock not supported in EM2 or EM3.

Hardware sets the interrupt flags in IADC_IF, and the flags remain set (sticky) until cleared by software. The interrupts flags should be cleared before enabling the IADC to remove any previous interrupt history. Clearing or setting interrupt bits can be done by writing to IADC_IF with a set or clear mask.

24.4 IADC Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	IADC_IPVERSION	R	IPVERSION
0x004	IADC_EN	RW ENABLE	Enable
0x008	IADC_CTRL	RW CONFIG	Control
0x00C	IADC_CMD	W SYNC	Command
0x010	IADC_TIMER	RW CONFIG	Timer
0x014	IADC_STATUS	RH	Status
0x018	IADC_MASKREQ	RW SYNC	Mask Request
0x01C	IADC_STMASK	RH SYNC	Scan Table Mask
0x020	IADC_CMPTHR	RW CONFIG	Digital Window Comparator Threshold
0x024	IADC_IF	RWH INTFLAG	Interrupt Flags
0x028	IADC_IEN	RW	Interrupt Enable
0x02C	IADC_TRIGGER	RW CONFIG	Trigger
0x048	IADC_CFGx	RW CONFIG	Configuration
0x050	IADC_SCALEx	RW CONFIG	Scaling
0x054	IADC_SCHEDx	RW CONFIG	Scheduling
0x070	IADC_SINGLEFIFOCFG	RW CONFIG	Single FIFO Configuration
0x074	IADC_SINGLEFIFODATA	R(r)H	Single FIFO Read Data
0x078	IADC_SINGLEFIFOSTAT	RH	Single FIFO Status
0x07C	IADC_SINGLEDATA	RH SYNC	Single Data
0x080	IADC_SCANFIFOCFG	RW CONFIG	Scan FIFO Configuration
0x084	IADC_SCANFIFODATA	R(r)H	Scan FIFO Read Data
0x088	IADC_SCANFIFOSTAT	RH	Scan FIFO Status
0x08C	IADC_SCANDATA	RH SYNC	Scan Data
0x098	IADC_SINGLE	RW SYNC	Single Queue Port Selection
0x0A0	IADC_SCANx	RW CONFIG	SCAN Entry
0x1000	IADC_IPVERSION_SET	R	IPVERSION
0x1004	IADC_EN_SET	RW ENABLE	Enable
0x1008	IADC_CTRL_SET	RW CONFIG	Control
0x100C	IADC_CMD_SET	W SYNC	Command
0x1010	IADC_TIMER_SET	RW CONFIG	Timer
0x1014	IADC_STATUS_SET	RH	Status
0x1018	IADC_MASKREQ_SET	RW SYNC	Mask Request
0x101C	IADC_STMASK_SET	RH SYNC	Scan Table Mask
0x1020	IADC_CMPTHR_SET	RW CONFIG	Digital Window Comparator Threshold
0x1024	IADC_IF_SET	RWH INTFLAG	Interrupt Flags

Offset	Name	Type	Description
0x1028	IADC_IEN_SET	RW	Interrupt Enable
0x102C	IADC_TRIGGER_SET	RW CONFIG	Trigger
0x1048	IADC_CFGx_SET	RW CONFIG	Configuration
0x1050	IADC_SCALEx_SET	RW CONFIG	Scaling
0x1054	IADC_SCHEDx_SET	RW CONFIG	Scheduling
0x1070	IADC_SINGLEFIFOCFG_SET	RW CONFIG	Single FIFO Configuration
0x1074	IADC_SINGLEFIFODATA_SET	R(r)H	Single FIFO Read Data
0x1078	IADC_SINGLEFIFOSTAT_SET	RH	Single FIFO Status
0x107C	IADC_SINGLEDATA_SET	RH SYNC	Single Data
0x1080	IADC_SCANFIFOCFG_SET	RW CONFIG	Scan FIFO Configuration
0x1084	IADC_SCANFIFODATA_SET	R(r)H	Scan FIFO Read Data
0x1088	IADC_SCANFIFOSTAT_SET	RH	Scan FIFO Status
0x108C	IADC_SCANDATA_SET	RH SYNC	Scan Data
0x1098	IADC_SINGLE_SET	RW SYNC	Single Queue Port Selection
0x10A0	IADC_SCANx_SET	RW CONFIG	SCAN Entry
0x2000	IADC_IPVERSION_CLR	R	IPVERSION
0x2004	IADC_EN_CLR	RW ENABLE	Enable
0x2008	IADC_CTRL_CLR	RW CONFIG	Control
0x200C	IADC_CMD_CLR	W SYNC	Command
0x2010	IADC_TIMER_CLR	RW CONFIG	Timer
0x2014	IADC_STATUS_CLR	RH	Status
0x2018	IADC_MASKREQ_CLR	RW SYNC	Mask Request
0x201C	IADC_STMASK_CLR	RH SYNC	Scan Table Mask
0x2020	IADC_CMPTHR_CLR	RW CONFIG	Digital Window Comparator Threshold
0x2024	IADC_IF_CLR	RWH INTFLAG	Interrupt Flags
0x2028	IADC_IEN_CLR	RW	Interrupt Enable
0x202C	IADC_TRIGGER_CLR	RW CONFIG	Trigger
0x2048	IADC_CFGx_CLR	RW CONFIG	Configuration
0x2050	IADC_SCALEx_CLR	RW CONFIG	Scaling
0x2054	IADC_SCHEDx_CLR	RW CONFIG	Scheduling
0x2070	IADC_SINGLEFIFOCFG_CLR	RW CONFIG	Single FIFO Configuration
0x2074	IADC_SINGLEFIFODATA_CLR	R(r)H	Single FIFO Read Data
0x2078	IADC_SINGLEFIFOSTAT_CLR	RH	Single FIFO Status
0x207C	IADC_SINGLEDATA_CLR	RH SYNC	Single Data
0x2080	IADC_SCANFIFOCFG_CLR	RW CONFIG	Scan FIFO Configuration
0x2084	IADC_SCANFIFODATA_CLR	R(r)H	Scan FIFO Read Data
0x2088	IADC_SCANFIFOSTAT_CLR	RH	Scan FIFO Status

Offset	Name	Type	Description
0x208C	IADC_SCANDATA_CLR	RH SYNC	Scan Data
0x2098	IADC_SINGLE_CLR	RW SYNC	Single Queue Port Selection
0x20A0	IADC_SCANx_CLR	RW CONFIG	SCAN Entry
0x3000	IADC_IPVERSION_TGL	R	IPVERSION
0x3004	IADC_EN_TGL	RW ENABLE	Enable
0x3008	IADC_CTRL_TGL	RW CONFIG	Control
0x300C	IADC_CMD_TGL	W SYNC	Command
0x3010	IADC_TIMER_TGL	RW CONFIG	Timer
0x3014	IADC_STATUS_TGL	RH	Status
0x3018	IADC_MASKREQ_TGL	RW SYNC	Mask Request
0x301C	IADC_STMASK_TGL	RH SYNC	Scan Table Mask
0x3020	IADC_CMPTHR_TGL	RW CONFIG	Digital Window Comparator Threshold
0x3024	IADC_IF_TGL	RWH INTFLAG	Interrupt Flags
0x3028	IADC_IEN_TGL	RW	Interrupt Enable
0x302C	IADC_TRIGGER_TGL	RW CONFIG	Trigger
0x3048	IADC_CFGx_TGL	RW CONFIG	Configuration
0x3050	IADC_SCALEx_TGL	RW CONFIG	Scaling
0x3054	IADC_SCHEDx_TGL	RW CONFIG	Scheduling
0x3070	IADC_SINGLEFIFOCFG_TGL	RW CONFIG	Single FIFO Configuration
0x3074	IADC_SINGLEFIFODATA_TGL	R(r)H	Single FIFO Read Data
0x3078	IADC_SINGLEFIFOSTAT_TGL	RH	Single FIFO Status
0x307C	IADC_SINGLEDATA_TGL	RH SYNC	Single Data
0x3080	IADC_SCANFIFOCFG_TGL	RW CONFIG	Scan FIFO Configuration
0x3084	IADC_SCANFIFODATA_TGL	R(r)H	Scan FIFO Read Data
0x3088	IADC_SCANFIFOSTAT_TGL	RH	Scan FIFO Status
0x308C	IADC_SCANDATA_TGL	RH SYNC	Scan Data
0x3098	IADC_SINGLE_TGL	RW SYNC	Single Queue Port Selection
0x30A0	IADC_SCANx_TGL	RW CONFIG	SCAN Entry

24.5 IADC Register Description

24.5.1 IADC_IPVERSION - IPVERSION

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x1	R	**IP version ID**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

24.5.2 IADC_EN - Enable

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**Enable IADC Module**
	The EN bit enables the module. Software should write to CONFIG type registers before setting the EN bit. Software should write to SYNC type registers only after setting the EN bit.			

Value	Mode	Description
0	DISABLE	Disable
1	ENABLE	Enable

24.5.3 IADC_CTRL - Control

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0								0x0																	0x0		0x0	0x0	0x0	0x0
Access		RW								RW																	RW		RW	RW	RW	RW
Name		HSCLKRATE								TIMEBASE																	WARMUPMODE		DBGHALT	ADCCLKSUSPEND1	ADCCLKSUSPEND0	EM23WUCONVERT

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:28	HSCLKRATE	0x0	RW	**High Speed Clock Rate**

Ratio to divide incoming CLK_CMU_ADC clock by. The resulting clock (CLK_SRC_ADC) must be 40 MHz or less.

Value	Mode	Description
0	DIV1	Use CMU_CLK_ADC directly. The source clock must be 40 MHz or less.
1	DIV2	Divide CMU_CLK_ADC by 2 before using it. The resulting CLK_SRC_ADC must be 40 MHz or less.
2	DIV3	Divide CMU_CLK_ADC by 3 before using it. The resulting CLK_SRC_ADC must be 40 MHz or less.
3	DIV4	Divide CMU_CLK_ADC by 4 before using it. The resulting CLK_SRC_ADC must be 40 MHz or less.

27:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:16	TIMEBASE	0x0	RW	**Time Base**

ADC clock cycles (TIMEBASE + 1) needed to generate a 1 us interval for warm up and start up timing. Does not allow less than 2 cycles. A setting of 0x0 (1 cycle) is replaced with 0x1 (2 cycles).

15:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	WARMUPMODE	0x0	RW	**Warmup Mode**

Select the warmup mode for the ADC.

Value	Mode	Description
0	NORMAL	Shut down the IADC after conversions have completed.
1	KEEPINSTANDBY	Switch to standby mode after conversions have completed. The next warmup time will require 1us.
2	KEEPWARM	Keep IADC fully powered after conversions have completed.

3	DBGHALT	0x0	RW	**Debug Halt**

Bit	Name	Reset	Access	Description
	ADC behavior when halted by debugger.			

Value	Mode	Description
0	NORMAL	Continue operation as normal during debug mode
1	HALT	Complete the current conversion and then halt during debug mode

Bit	Name	Reset	Access	Description
2	ADCCLKSUSPEND1	0x0	RW	**ADC_CLK Suspend - PRS1**
	This only functions with single trigger select set to PRSPOS or PRSNEG. In EM0 and EM1, this gates the local clock while clock source remains running. In EM2 and EM3, this disables the clock source until the PRSPOS or PRSNEG event is detected. This bit has no effect if the local IADC timer is running.			

Value	Mode	Description
0	PRSWUDIS	Normal mode which does not disable the ADC_CLK.
1	PRSWUEN	ADCCLKWUEN will gate off ADC_CLK until the trigger is detected provided the internal timer is not selected as the trigger. Once the trigger is detected the ADC_CLK will be started, the band gap will be started, the ADC will be warmed up, and the SCAN Table and the Single entry will be converted. Once the conversions are done, the ADC_CLK will be gated off.

Bit	Name	Reset	Access	Description
1	ADCCLKSUSPEND0	0x0	RW	**ADC_CLK Suspend - PRS0**
	This only functions with scan trigger select set to PRSPOS or PRSNEG. In EM0 and EM1, this gates the local clock while clock source remains running. In EM2 and EM3, this disables the clock source until the PRSPOS or PRSNEG event is detected. This bit has no effect if the local IADC timer is running.			

Value	Mode	Description
0	PRSWUDIS	Normal mode which does not disable the ADC_CLK.
1	PRSWUEN	ADCCLKWUEN will gate off ADC_CLK until the trigger is detected provided the internal timer is not selected as the trigger. Once the trigger is detected the ADC_CLK will be started, the band gap will be started, the ADC will be warmed up, and the SCAN Table and the Single entry will be converted. Once the conversions are done, the ADC_CLK will be gated off.

Bit	Name	Reset	Access	Description
0	EM23WUCONVERT	0x0	RW	**EM23 Wakeup on Conversion**
	EM23 wake up on conversion			

Value	Mode	Description
0	WUDVL	When using suspend mode, conversions performed in EM2 or EM3 should not wake up the DMA until the FIFO's DVL setting is reached. This saves more power for large OSR settings or infrequent sampling.
1	WUCONVERT	When using suspend mode, conversions performed in EM2 or EM3 will wake up the DMA and keep it awake until the conversions are done, regardless of the DVL setting. This mode burns more power, but it is useful when the conversion rate is faster than the time for the DMA to cycle through wake up and going back to sleep as it converts more than 4 scan table entries. Without using the wake up on conversion mode, the FIFO may overflow while the DMA is going in and out of sleep.

24.5.4 IADC_CMD - Command

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset							0x0	0x0							0x0	0x0												0x0	0x0		0x0	0x0
Access							W	W							W	W												W	W		W	W
Name							SCANFIFOFLUSH	SINGLEFIFOFLUSH							TIMERDIS	TIMEREN												SCANSTOP	SCANSTART		SINGLESTOP	SINGLESTART

Bit	Name	Reset	Access	Description
31:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	SCANFIFOFLUSH	0x0	W	**Flush the Scan FIFO**
	Flush the Scan FIFO. The IADC must be enabled, not suspended, and the IADC clock must be running. Operation has completed when STATUS.SCANFIFOFLUSHING has gone low. The scan queue should be disabled. Any incoming scan queue data will be discarded during the flush.			
24	SINGLEFIFOFLUSH	0x0	W	**Flush the Single FIFO**
	Flush the Single FIFO. The IADC must be enabled, not suspended, and the IADC clock must be running. Operation has completed when STATUS.SINGLEFIFOFLUSHING has gone low. The Single queue should be disabled. Any incoming single queue data will be discarded during the flush.			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	TIMERDIS	0x0	W	**Timer Disable**
	Disable the local timer and reset the counter to timer reload value.			
16	TIMEREN	0x0	W	**Timer Enable**
	Enable the local timer.			
15:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	SCANSTOP	0x0	W	**Scan Queue Stop**
	Stop the Scan queue. Disables Scan triggers and clears pending conversions in the Scan queue. Any conversion that has already started will continue until it is complete. If the scan queue is stopped before all entries of the scan table have completed, the remaining entries will not be converted.			
3	SCANSTART	0x0	W	**Scan Queue Start**
	Start the Scan queue. Enables triggering of the Scan queue.			
2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	SINGLESTOP	0x0	W	**Single Queue Stop**
	Stop the Single queue. Disables Single queue triggers and clears pending conversions in the Single queue. Any conversion that has already started will continue until it is complete.			
0	SINGLESTART	0x0	W	**Single Queue Start**
	Start the Single queue. Enables triggering of the Single queue.			

24.5.5 IADC_TIMER - Timer

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	RW															
Name																	TIMER															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	TIMER	0x0	RW	**Timer Period**
	Number of CLK_SRC_ADC cycles per timer event.			

24.5.6 IADC_STATUS - Status

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0						0x0			0x0	0x0				0x0	0x0	0x0					0x0	0x0		0x0		0x0	0x0		0x0	0x0
Access		R						R			R	R				R	R	R					R	R		R		R	R		R	R
Name		ADCWARM						SYNCBUSY			MASKREQWRITEPENDING	SINGLEWRITEPENDING				TIMERACTIVE	SCANFIFOFLUSHING	SINGLEFIFOFLUSHING					SCANFIFODV	SINGLEFIFODV		CONVERTING		SCANQUEUEPENDING	SCANQEN		SINGLEQUEUEPENDING	SINGLEQEN

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30	ADCWARM	0x0	R	**ADCWARM**
	The ADC analog front end and reference require a delay before converting when coming from a powered down or standby state. This status bit indicates that the analog front end and reference are ready.			
29:25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	SYNCBUSY	0x0	R	**SYNCBUSY**
	Indicates synchronization ongoing			
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	MASKREQWRITEPENDING	0x0	R	**MASKREQ write pending**
	A write to MASKREQ is pending. The ADC converts using a local working mask register, and only transfers MASKREQ to the local working version when the SCAN queue is not converting.			
20	SINGLEWRITEPENDING	0x0	R	**SINGLE write pending**
	The SINGLE register write is pending. The ADC converts using a local working version of the SINGLE register, and only transfer SINGLE to the local working version when the SINGLE queue is not being converted.			
19:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	TIMERACTIVE	0x0	R	**Timer Active**
	The local timer is running.			
15	SCANFIFOFLUSHING	0x0	R	**The Scan FIFO is flushing**
	A scan data FIFO flush operation is in progress.			
14	SINGLEFIFOFLUSHING	0x0	R	**The Single FIFO is flushing**
	A single data FIFO flush operation is in progress.			

Bit	Name	Reset	Access	Description
13:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	SCANFIFODV	0x0	R	**SCANFIFO Data Valid**
	At least one result in the single FIFO is ready to read.			
8	SINGLEFIFODV	0x0	R	**SINGLEFIFO Data Valid**
	At least one result in the scan FIFO is ready to read.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	CONVERTING	0x0	R	**Converting**
	The ADC is warmed up and in the process of performing a conversion.			
5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	SCANQUEUEPENDING	0x0	R	**Scan Queue Pending**
	The Scan queue has been triggered and is waiting to start conversion.			
3	SCANQEN	0x0	R	**Scan Queued Enabled**
	The Scan queue is enabled.			
2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	SINGLEQUEUEPEND-ING	0x0	R	**Single Queue Pending**
	The Single queue has been triggered and is waiting to start conversion. When tailgating is used, SINGLEQUEUEPEND-ING will remain high until the a scan operation has completed.			
0	SINGLEQEN	0x0	R	**Single Queue Enabled**
	The Single queue is enabled.			

24.5.7 IADC_MASKREQ - Mask Request

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	RW															
Name																	MASKREQ															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	MASKREQ	0x0	RW	**Scan Queue Mask Request**
	Allows software to specify which entries in the Scan table should be converted. For example MASKREQ = 0x8014 means that scan table entries 15, 4, and 2 will be converted. The other entries will not be converted.			

24.5.8 IADC_STMASK - Scan Table Mask

Offset	Bit Position																															
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0															
Access																	R															
Name																	STMASK															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	STMASK	0x0	R	**Scan Table Mask**
	This is the active / working copy of the MASKREQ register that the ADC uses. It will only be updated at the end of a scan sequence or when no scan is in progress.			

24.5.9 IADC_CMPTHR - Digital Window Comparator Threshold

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0x0									0x0															
Access								RW									RW															
Name								ADGT									ADLT															

Bit	Name	Reset	Access	Description
31:16	ADGT	0x0	RW	**ADC Greater Than or Equal to Threshold**
	Compare threshold value for greater-than or equal to comparison. ADGT should be specified in a left-justified, 16-bit format regardless of the FIFO ALIGNMENT setting. Comparisons with 12-bit formats will ignore the 4 LSBs of the ADGT value. Comparisons with 20-bit formats will ignore the 4 LSBs of the 20-bit result. Unipolar or bipolar mode is considered in the comparison. When ADGT is greater than ADLT, the comparison is true if the result is either greater than ADGT or less than ADLT, but false if the result falls between the values.			
15:0	ADLT	0x0	RW	**ADC Less Than or Equal to Threshold**
	Compare threshold value for less-than or equal to comparison. ADLT should be specified in a left-justified, 16-bit format regardless of the FIFO ALIGNMENT setting. Comparisons with 12-bit formats will ignore the 4 LSBs of the ADLT value. Comparisons with 20-bit formats will ignore the 4 LSBs of the 20-bit result. Unipolar or bipolar mode is considered in the comparison. When ADGT is greater than ADLT, the comparison is true if the result is either greater than ADGT or less than ADLT, but false if the result falls between the values.			

24.5.10 IADC_IF - Interrupt Flags

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0												0x0	0x0	0x0	0x0			0x0	0x0			0x0	0x0	0x0				0x0	0x0	0x0	0x0
Access	RW												RW	RW	RW	RW			RW	RW			RW	RW	RW				RW	RW	RW	RW
Name	EM23ABORTERROR												SCANFIFOUF	SINGLEFIFOUF	SCANFIFOOF	SINGLEFIFOOF			PORTALLOCERR	POLARITYERR			SINGLEDONE	SCANTABLEDONE	SCANENTRYDONE				SCANCMP	SINGLECMP	SCANFIFODVL	SINGLEFIFODVL

Bit	Name	Reset	Access	Description
31	EM23ABORTERROR	0x0	RW	**EM2/3 Abort Error**
	The system entered EM2 or EM3 during a conversion with an unsupported clock. Conversion results may be corrupted.			
30:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	SCANFIFOUF	0x0	RW	**Scan FIFO Underflow**
	A scan FIFO underflow has occured.			
18	SINGLEFIFOUF	0x0	RW	**Single FIFO Underflow**
	A single FIFO underflow has occured.			
17	SCANFIFOOF	0x0	RW	**Scan FIFO Overflow**
	A scan FIFO overflow has occured.			
16	SINGLEFIFOOF	0x0	RW	**Single FIFO Overflow**
	A single FIFO overflow has occured.			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	PORTALLOCERR	0x0	RW	**Port Allocation Error**
	A pin was selected on a port which has not been allocated to the IADC in GPIO control.			
12	POLARITYERR	0x0	RW	**Polarity Error**
	Either two even channels or two odd channels were programmed into the channel mux selection. The ADC result will be set to 0xFFFF.			
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	SINGLEDONE	0x0	RW	**Single Conversion Done**
	A single conversion has completed.			
8	SCANTABLEDONE	0x0	RW	**Scan Table Done**
	A scan sequence completed. Set at the end of a scan sequence after all valid entries of the scan table have completed.			
7	SCANENTRYDONE	0x0	RW	**Scan Entry Done**
	A scan table conversion completed. Set at the completion of each valid entry of the scan table.			

Bit	Name	Reset	Access	Description
6:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	SCANCMP	0x0	RW	**Scan Result Window Compare**
	Scan digital compare window tripped.			
2	SINGLECMP	0x0	RW	**Single Result Window Compare**
	Single digital compare window tripped.			
1	SCANFIFODVL	0x0	RW	**Scan FIFO Data Valid Level**
	A minimum of (DVL+1) entries are ready to be read from the Scan FIFO.			
0	SINGLEFIFODVL	0x0	RW	**Single FIFO Data Valid Level**
	A minimum of (DVL+1) entries are ready to be read from the Single FIFO.			

24.5.11 IADC_IEN - Interrupt Enable

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0												0x0	0x0	0x0	0x0			0x0	0x0			0x0	0x0	0x0				0x0	0x0	0x0	0x0
Access	RW												RW	RW	RW	RW			RW	RW			RW	RW	RW				RW	RW	RW	RW
Name	EM23ABORTERROR												SCANFIFOUF	SINGLEFIFOUF	SCANFIFOOF	SINGLEFIFOOF			PORTALLOCERR	POLARITYERR			SINGLEDONE	SCANTABLEDONE	SCANENTRYDONE				SCANCMP	SINGLECMP	SCANFIFODVL	SINGLEFIFODVL

Bit	Name	Reset	Access	Description
31	EM23ABORTERROR	0x0	RW	**EM2/3 Abort Error Enable**
	EM2/3 Abort Error Enable			
30:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	SCANFIFOUF	0x0	RW	**Scan FIFO Underflow Enable**
	Scan FIFO underflow interrupt enable			
18	SINGLEFIFOUF	0x0	RW	**Single FIFO Underflow Enable**
	Single FIFO underflow interrupt enable			
17	SCANFIFOOF	0x0	RW	**Scan FIFO Overflow Enable**
	Scan FIFO overflow interrupt enable			
16	SINGLEFIFOOF	0x0	RW	**Single FIFO Overflow Enable**
	Single FIFO overflow interrupt enable			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	PORTALLOCERR	0x0	RW	**Port Allocation Error Enable**
	Port Allocation Error Enable			
12	POLARITYERR	0x0	RW	**Polarity Error Enable**
	Polarity Error Enable			
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	SINGLEDONE	0x0	RW	**Single Conversion Done Enable**
	Single Conversion Done interrupt enable			
8	SCANTABLEDONE	0x0	RW	**Scan Table Done Enable**
	Scan Table Done interrupt enable			
7	SCANENTRYDONE	0x0	RW	**Scan Entry Done Enable**
	Scan Entry Done interrupt enable			

Bit	Name	Reset	Access	Description
6:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	SCANCMP	0x0	RW	**Scan Result Window Compare Enable**
	Scan Result Window Compare Enable			
2	SINGLECMP	0x0	RW	**Single Result Window Compare Enable**
	Single Result Window Compare Enable			
1	SCANFIFODVL	0x0	RW	**Scan FIFO Data Valid Level Enable**
	Scan FIFO Data Valid Level interrupt enable			
0	SINGLEFIFODVL	0x0	RW	**Single FIFO Data Valid Level Enable**
	Single FIFO Data Valid Level interrupt enable			

24.5.12 IADC_TRIGGER - Trigger

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0x0				0x0			0x0					0x0			0x0	
Access																RW				RW			RW					RW			RW	
Name																SINGLETAILGATE				SINGLETRIGACTION			SINGLETRIGSEL					SCANTRIGACTION			SCANTRIGSEL	

Bit	Name	Reset	Access	Description
31:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	SINGLETAILGATE	0x0	RW	**Single Tailgate Enable**
	Enables tailgating.			
	Value	Mode		Description
	0	TAILGATEOFF		The single queue is ready to start warming up and converting once the trigger had been detected.
	1	TAILGATEON		After the single queue's trigger is detected, it must wait until the end of a scan operation before the Single queue can be converted.
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	SINGLETRIGACTION	0x0	RW	**Single Trigger Action**
	Selects the trigger action for the single queue.			
	Value	Mode		Description
	0	ONCE		For TRIGSEL=IMMEDIATE, converts the single queue once and disables queue. For TRIGSEL = TIMER, PRSCLKGRP, PRSPOS, PRSNEG, converts the single queue once per trigger.ask.
	1	CONTINUOUS		Converts the single queue, then checks for a pending scan queue before converting the single queue again continuously. The queues are first come first serve. If both queues are continuous, the IADC alternates between them.
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	SINGLETRIGSEL	0x0	RW	**Single Trigger Select**
	Selects the trigger source for the single queue.			
	Value	Mode		Description

Bit	Name	Reset	Access	Description
	0	IMMEDIATE		Immediate triggering. The single queue will be disabled once the conversion is complete, unless TRIGGERACTION is set to continuous.
	1	TIMER		Triggers when the local timer count reaches zero.
	2	PRSCLKGRP		Triggers on PRS1 from a timer module that is using the same clock group as the ADC and has been programmed to use the same clock source as the ADC. The prescale may be different between the ADC and the timer module.
	3	PRSPOS		Triggers on asynchronous PRS1 positive edge. Requires PRS1 to go low for 3 ADC_CLKs before another positive edge can be detected. Generates an additional delay of 1 to 2 ADC_SRC_CLK cycles for synchronization.
	4	PRSNEG		Triggers on asynchronous PRS1 negative edge. Requires PRS1 to go high for 3 ADC_CLKs before another negative edge can be detected. Generates an additional delay of 1 to 2 ADC_SRC_CLK cycles for synchronization. PRSNEG should only be used when the trigger source is from a module that remains powered during EM23. For modules (ie: TIMER) that power down during EM23, PRSPOS should be used for an asynchronous trigger, and PRSCLKGRP should be used for a synchronous trigger.
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	SCANTRIGACTION	0x0	RW	**Scan Trigger Action**
	Selects the trigger action for the scan queue.			
	Value	Mode		Description
	0	ONCE		For TRIGSEL=IMMEDIATE, goes through the scan table once and disables queue. For TRIGSEL = TIMER, PRSCLKGRP, PRSPOS, PRSNEG, goes through the scan table once per trigger.
	1	CONTINUOUS		Goes through the scan table, converts each entry with a mask bit set, and puts it back into the scan queue to repeat again continuously. The queues are first come first serve. If both queues are triggered, the single queue will get to convert after each scan table completes. The scan queue will get to convert after each single conversion completes.
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	SCANTRIGSEL	0x0	RW	**Scan Trigger Select**
	Selects the trigger source for the scan queue.			
	Value	Mode		Description
	0	IMMEDIATE		Immediate triggering. The scan queue will be disabled once all conversions in the scan table are complete, unless TRIGGERACTION is set to continuous.
	1	TIMER		Triggers when the local timer count reaches zero.
	2	PRSCLKGRP		Triggers on PRS0 from a timer module that is using the same clock group as the ADC and has been programmed to use the same clock source as the ADC. The prescale may be different between the ADC and the timer module.

Bit	Name	Reset	Access	Description
	3	PRSPOS		Triggers on asynchronous PRS0 positive edge. Requires PRS0 to go low for 3 ADC_CLKs before another positive edge can be detected. Generates an additional delay of 1 to 2 ADC_SRC_CLK cycles for synchronization.
	4	PRSNEG		Triggers on asynchronous PRS0 negative edge. Requires PRS0 to go high for 3 ADC_CLKs before another negative edge can be detected. Generates an additional delay of 1 to 2 ADC_SRC_CLK cycles for synchronization. PRSNEG should only be used when the trigger source is from a module that remains powered during EM23. For modules (ie: TIMER) that power down during EM23, PRSPOS should be used for an asynchronous trigger, and PRSCLKGRP should be used for a synchronous trigger.

24.5.13 IADC_CFGx - Configuration

Offset	Bit Position																													
0x048	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x0						0x0					0x0				0x2										0x0			0x0	
Access			RW						RW					RW				RW										RW			RW	
Name			TWOSCOMPL						DIGAVG					REFSEL				ANALOGGAIN										OSRHS			ADCMODE	

Bit	Name	Reset	Access	Description
31:30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29:28	TWOSCOMPL	0x0	RW	**Two's Complement**
	Selects output word polarity.			

Value	Mode	Description
0	AUTO	Automatic: Single ended measurements are reported as unipolar and differential measurements are reported as bipolar.
1	FORCEUNIPOLAR	Force all measurements to result in unipolar output. Negative differential numbers will saturate to 0.
2	FORCEBIPOLAR	Force all measurements to result in bipolar output. Single ended measurements are half the range, but allow for small negative measurements.

27:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:21	DIGAVG	0x0	RW	**Digital Averaging**
	Number of output words to convert and average.			

Value	Mode	Description
0	AVG1	Collect one output word (no digital averaging).
1	AVG2	Collect and average 2 digital output words.
2	AVG4	Collect and average 4 digital output words.
3	AVG8	Collect and average 8 digital output words.
4	AVG16	Collect and average 16 digital output words.

20:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	REFSEL	0x0	RW	**Reference Select**
	Selects voltage reference.			

Value	Mode	Description
0	VBGR	Internal 1.21 V reference.

Bit	Name	Reset	Access	Description
	1	VREF		External Reference. (Calibrated for 1.25V nominal.)
	3	VDDX		AVDD (unbuffered)
	4	VDDX0P8BUF		AVDD (buffered) * 0.8
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:12	ANALOGGAIN	0x2	RW	**Analog Gain**
	Sets analog froont end gain.			
	Value	Mode		Description
	1	ANAGAIN0P5		Analog gain of 0.5x.
	2	ANAGAIN1		Analog gain of 1x.
	3	ANAGAIN2		Analog gain of 2x.
	4	ANAGAIN3		Analog gain of 3x.
	5	ANAGAIN4		Analog gain of 4x.
11:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4:2	OSRHS	0x0	RW	**High Speed OSR**
	Over sampling ratio for high speed conversions.			
	Value	Mode		Description
	0	HISPD2		High speed over sampling of 2x.
	1	HISPD4		High speed over sampling of 4x.
	2	HISPD8		High speed over sampling of 8x.
	3	HISPD16		High speed over sampling of 16x.
	4	HISPD32		HIgh speed over sampling of 32x.
	5	HISPD64		High speed over sampling of 64x.
1:0	ADCMODE	0x0	RW	**ADC Mode**
	Selects ADC conversion mode.			
	Value	Mode		Description
	0	NORMAL		High speed mode with a maximum CLK_ADC of 10 MHz.

24.5.14 IADC_SCALEx - Scaling

Offset	Bit Position																													
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x1	0x0													0x2C000															
Access	RW	RW													RW															
Name	GAIN3MSB	GAIN13LSB													OFFSET															

Bit	Name	Reset	Access	Description
31	GAIN3MSB	0x1	RW	**Gain 3 MSBs**
	3 MSBs of the 16-bit gain value (0=011 or 0.75; 1=1xx or 1.00). Example {GAIN3MSB, GAIN13LSB} = {100, 0_1001_0000_0000} = 1.07031x. Example {GAIN3MSB, GAIN13LSB} = {011, 0_0000_1010_0010} = 0.75494x.			

Value	Mode	Description
0	GAIN011	Upper 3 bits of gain = 011 (0.75x)
1	GAIN100	Upper 3 bits of gain = 100 (1.00x)

Bit	Name	Reset	Access	Description
30:18	GAIN13LSB	0x0	RW	**Gain 13 LSBs**
	13 LSBs of the 16-bit gain value.			
17:0	OFFSET	0x2C000	RW	**Offset**
	Offset			

24.5.15 IADC_SCHEDx - Scheduling

Offset	Bit Position																													
0x054	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0x0								
Access																							RW								
Name																							PRESCALE								

Bit	Name	Reset	Access	Description
31:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:0	PRESCALE	0x0	RW	**Prescale**
	Second level prescaler - divides the CLK_SRC_ADC by (PRESCALE + 1) to generate CLK_ADC. PRESCALE=0 should only be used with HSCLKRATE=0. (See text.)			

24.5.16 IADC_SINGLEFIFOCFG - Single FIFO Configuration

Offset	Bit Position																														
0x070	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0			0x3		0x0		0x0	
Access																								RW			RW		RW		RW	
Name																								DMAWUFIFOSINGLE			DVL		SHOWID		ALIGNMENT	

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	DMAWUFIFOSINGLE	0x0	RW	**Single FIFO DMA wakeup.**

Enables single FIFO to wake DMA in EM2 or EM3.

Value	Mode	Description
0	DISABLED	While in EM2 or EM3, the DMA controller will not be requested.
1	ENABLED	While in EM2 or EM3, the DMA controller will be requested when the single FIFO reaches its Data Valid Level. [DVL must be set to 0 (VALID1).]

Bit	Name	Reset	Access	Description
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	DVL	0x3	RW	**Data Valid Level**

Data valid level before requesting DMA transfer. If the number of words in the FIFO reaches or exceeds DVL+1, DMA requests will be generated.

Value	Mode	Description
0	VALID1	When 1 entry in the single FIFO is valid, set the SINGLEFIFODVL interrupt and request DMA.
1	VALID2	When 2 entries in the single FIFO are valid, set the SINGLEFIFODVL interrupt and request DMA.
2	VALID3	When 3 entries in the single FIFO are valid, set the SINGLEFIFODVL interrupt and request DMA.
3	VALID4	When 4 entries in the single FIFO are valid, set the SINGLEFIFODVL interrupt and request DMA.

Bit	Name	Reset	Access	Description
3	SHOWID	0x0	RW	**Show ID**

ID of 0x20 will be applied in the output words.

Bit	Name	Reset	Access	Description
2:0	ALIGNMENT	0x0	RW	**Alignment**

Alignment of output data written into FIFO.

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	RIGHT12		ID[7:0], SIGN_EXT, DATA[11:0]
	1	RIGHT16		ID[7:0], SIGN_EXT, DATA[15:0]
	2	RIGHT20		ID[7:0], SIGN_EXT, DATA[19:0]
	3	LEFT12		DATA[11:0], 000000000000, ID[7:0]
	4	LEFT16		DATA[15:0], 00000000, ID[7:0]
	5	LEFT20		DATA[19:0], 0000, ID[7:0]

24.5.17 IADC_SINGLEFIFODATA - Single FIFO Read Data

Offset	Bit Position																													
0x074	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R(r)																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x0	R(r)	**Single FIFO Read Data**
	Reads and pops the oldest value from the single FIFO.			

24.5.18 IADC_SINGLEFIFOSTAT - Single FIFO Status

Offset	Bit Position																													
0x078	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0		
Access																														R		
Name																														FIFOREADCNT		

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	FIFOREADCNT	0x0	R	**FIFO Read Count**
	Number of valid entries available to read.			

24.5.19 IADC_SINGLEDATA - Single Data

Offset	Bit Position																													
0x07C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x0	R	**Data**
	Reads the most recent data word from the single FIFO, but does not pop a value. Even if the FIFO has overflowed and stopped updating, the most recent conversion will continue to overwrite SINGLEDATA.			

24.5.20 IADC_SCANFIFOCFG - Scan FIFO Configuration

Offset	Bit Position																														
0x080	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0x0			0x3		0x0		0x0	
Access																								RW			RW		RW		RW	
Name																								DMAWUFIFOSCAN			DVL		SHOWID		ALIGNMENT	

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	DMAWUFIFOSCAN	0x0	RW	**Scan FIFO DMA Wakeup**
	Enables scan FIFO to wake DMA in EM2 or EM3.			
	Value	Mode		Description
	0	DISABLED		While in EM2 or EM3, the DMA controller will not be requested.
	1	ENABLED		While in EM2 or EM3, the DMA controller will be requested when the scan FIFO reaches its Data Valid Level. [DVL must be set to 0 (VALID1).]
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	DVL	0x3	RW	**Data Valid Level**
	Data valid level before requesting DMA transfer. If the number of words in the FIFO reaches or exceeds DVL+1, DMA requests will be generated.			
	Value	Mode		Description
	0	VALID1		When 1 entry in the scan FIFO is valid, set the SCANFIFODVL interrupt and request DMA.
	1	VALID2		When 2 entries in the scan FIFO are valid, set the SCANFIFODVL interrupt and request DMA.
	2	VALID3		When 3 entries in the scan FIFO are valid, set the SCANFIFODVL interrupt and request DMA.
	3	VALID4		When 4 entries in the scan FIFO are valid, set the SCANFIFODVL interrupt and request DMA.
3	SHOWID	0x0	RW	**Show ID**
	Enable ID in output words.			
2:0	ALIGNMENT	0x0	RW	**Alignment**
	Alignment of output data written into FIFO.			
	Value	Mode		Description

Bit	Name	Reset	Access	Description
	0	RIGHT12		ID[7:0], SIGN_EXT, DATA[11:0]
	1	RIGHT16		ID[7:0], SIGN_EXT, DATA[15:0]
	2	RIGHT20		ID[7:0], SIGN_EXT, DATA[19:0]
	3	LEFT12		DATA[11:0], 000000000000, ID[7:0]
	4	LEFT16		DATA[15:0], 00000000, ID[7:0]
	5	LEFT20		DATA[19:0], 0000, ID[7:0]

24.5.21 IADC_SCANFIFODATA - Scan FIFO Read Data

Offset	Bit Position																														
0x084	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R(r)																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x0	R(r)	**Data**
	Reads and pops the oldest value from the scan FIFO.			

24.5.22 IADC_SCANFIFOSTAT - Scan FIFO Status

Offset	Bit Position																														
0x088	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0		
Access																														R		
Name																														FIFOREADCNT		

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	FIFOREADCNT	0x0	R	**FIFO Read Count**
	Number of valid entries available to read.			

24.5.23 IADC_SCANDATA - Scan Data

Offset	Bit Position																													
0x08C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x0	R	**Data**
	Reads the most recent data word from the scan FIFO, but does not pop a value. Even if the FIFO has overflowed and stopped updating, the most recent conversion will continue to overwrite SCANDATA.			

24.5.24 IADC_SINGLE - Single Queue Port Selection

Offset	Bit Position																													
0x098	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0	0x0	0x0				0x0				0x0				0x0			
Access															RW	RW	RW				RW				RW				RW			
Name															CMP	CFG	PORTPOS				PINPOS				PORTNEG				PINNEG			

Bit	Name	Reset	Access	Description
31:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	CMP	0x0	RW	**Comparison Enable**
	Enable digital window comparison for this entry.			
16	CFG	0x0	RW	**Configuration Group Select**
	Select which configuration group (CFGx, SCALEx, SCHEDx registers) is used with this entry.			

Value	Mode	Description
0	CONFIG0	Use configuration group 0
1	CONFIG1	Use configuration group 1

Bit	Name	Reset	Access	Description
15:12	PORTPOS	0x0	RW	**Positive Port Select**
	Port (A, B, C, D) or special signal assigned to the positve input of the ADC			

Value	Mode	Description
0	GND	Ground
1	SUPPLY	Supply Pin - Select specific supply using PINPOS
8	PORTA	Port A - Select pin number using PINPOS
9	PORTB	Port B - Select pin number using PINPOS
10	PORTC	Port C - Select pin number using PINPOS
11	PORTD	Port D - Select pin number using PINPOS

Bit	Name	Reset	Access	Description
11:8	PINPOS	0x0	RW	**Positive Pin Select**
	Pin number for the positive input of the ADC.			
7:4	PORTNEG	0x0	RW	**Negative Port Select**
	Port (A, B, C, D) or special signal assigned to the negative input of the ADC			

Value	Mode	Description
0	GND	Ground (single-ended)
8	PORTA	Port A - Select pin number using PINNEG
9	PORTB	Port B - Select pin number using PINNEG
10	PORTC	Port C - Select pin number using PINNEG

Bit	Name	Reset	Access	Description
	11	PORTD		Port D - Select pin number using PINNEG
3:0	PINNEG	0x0	RW	**Negative Pin Select**
	Pin number for the negative input of the ADC.			

24.5.25 IADC_SCANx - SCAN Entry

Offset	Bit Position																													
0x0A0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0	0x0	0x0				0x0				0x0				0x0			
Access															RW	RW	RW				RW				RW				RW			
Name															CMP	CFG	PORTPOS				PINPOS				PORTNEG				PINNEG			

Bit	Name	Reset	Access	Description
31:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	CMP	0x0	RW	**Comparison Enable**
	Enable digital window comparison for this entry.			
16	CFG	0x0	RW	**Configuration Group Select**
	Select which configuration group (CFGx, SCALEx, SCHEDx registers) is used with this entry.			

Value	Mode	Description
0	CONFIG0	Use configuration group 0
1	CONFIG1	Use configuration group 1

Bit	Name	Reset	Access	Description
15:12	PORTPOS	0x0	RW	**Positive Port Select**
	Port (A, B, C, D) or special signal assigned to the positve input of the ADC			

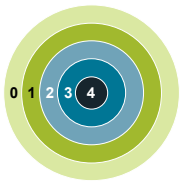
Value	Mode	Description
0	GND	Ground
1	SUPPLY	Supply Pin - Select specific supply using PINPOS
8	PORTA	Port A - Select pin number using PINPOS
9	PORTB	Port B - Select pin number using PINPOS
10	PORTC	Port C - Select pin number using PINPOS
11	PORTD	Port D - Select pin number using PINPOS

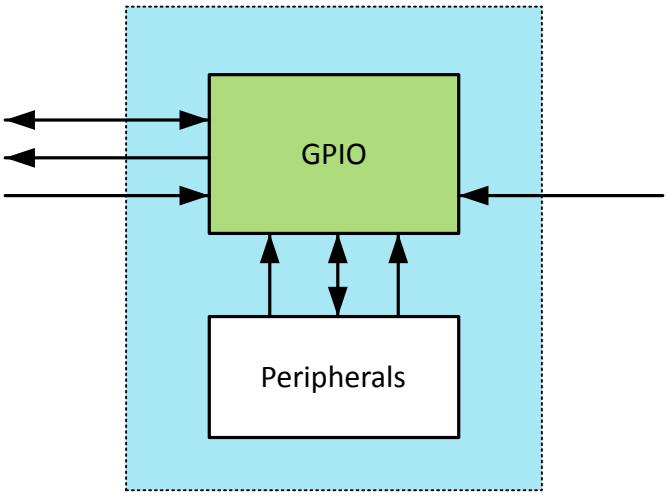
Bit	Name	Reset	Access	Description
11:8	PINPOS	0x0	RW	**Positive Pin Select**
	Pin number for the positive input of the ADC.			
7:4	PORTNEG	0x0	RW	**Negative Port Select**
	Port (A, B, C, D) or special signal assigned to the negative input of the ADC			

Value	Mode	Description
0	GND	Ground (single-ended)
8	PORTA	Port A - Select pin number using PINNEG
9	PORTB	Port B - Select pin number using PINNEG
10	PORTC	Port C - Select pin number using PINNEG

Bit	Name	Reset	Access	Description
	11	PORTD		Port D - Select pin number using PINNEG
3:0	PINNEG	0x0	RW	**Negative Pin Select**
	Pin number for the negative input of the ADC.			

25. GPIO - General Purpose Input/Output





Quick Facts

What?

The General Purpose Input/Output (GPIO) is used for pin configuration, direct pin manipulation, and sensing, as well as routing for peripheral pin connections.

Why?

Easy to use and highly configurable input/output pins are important to fit many communication protocols as well as minimizing software control overhead. Flexible routing of peripheral functions helps to ease PCB layout.

How?

Each pin on the device can be individually configured as either an input or an output with several different drive modes. Also, individual bit manipulation registers minimizes control overhead. Peripheral connections to pins can be routed to pins as desired solving congestion and contention issues that may arise with limited routing flexibility. Fully asynchronous interrupts can also be generated from any pin.

25.1 Introduction

In the EFR32xG22 devices the General Purpose Input/Output (GPIO) pins are organized into ports with up to 16 pins each. These GPIO pins can be individually configured as either an output or input. More advanced configurations like open-drain, open-source, and glitch filtering can be configured for each individual GPIO pin. Peripheral resources, like Timer PWM outputs or USART RX/TX can be routed to the GPIO pins as desired by the user. Finally, the input value of a pin can be routed through the Peripheral Reflex System to other peripherals or used to trigger an external interrupt.

25.2 Features

- Individual configuration for each pin
  - Tristate (reset state)
  - Push-pull
  - Open-drain
  - Pull-up resistor
  - Pull-down resistor
  - Programable Slewrate Control
- EM4 IO pin retention
  - Output enable
  - Output value
  - Pull enable
  - Pull direction
- EM4 wake-up on selected GPIO pins
- Glitch suppression input filter
- Extremely flexible analog and digital resource routing
- Toggle register for output data
- Dedicated data input register (read-only)
- Interrupts
  - 2 Interrupt lines using either levels or edges
    - EM4 wake-up pins are selectable for level interrupts
    - All GPIO pins are selectable for edge interrupts
  - Separate enable, status, set and clear registers
  - Asynchronous sensing
  - Rising, falling or both edges
  - High or low level detection
  - Wake up from EM1-EM3
- Peripheral Reflex System producer
  - All GPIO pins are selectable

25.3 Functional Description

An overview of the GPIO module is shown in Figure 25.1 Pin Configuration on page 779. The GPIO pins are grouped into 16-pin ports. Each individual GPIO pin is called Pxn where x indicates the port (A, B, C ...) and n indicates the pin number (0,1,....,15). Fewer than 16 pins may be available on some ports depending on the total number of I/O pins on the package. After a reset, both input and output are disabled for all pins on the device, except for the Serial Wire Debug pins.

To use a pin, the Mode Register (GPIO_Px_MODEL/GPIO_Px_MODEH) must be configured for the pin to make it an input or output. These registers can also do more advanced configuration, which is covered in 25.3.1 Pin Configuration. When the port is configured as an input or an output, the Data In Register (GPIO_Px_DIN) can be used to read the level of each pin in the port (bit n in the register is connected to pin n on the port). When configured as an output, the value of the Data Out Register (GPIO_Px_DOUT) will be driven to the pin.

The DOUT value can be changed in 4 different ways:

- Writing to the GPIO_Px_DOUT register
- Writing the SET address of the GPIO_Px_DOUT register sets the DOUT bits
- Writing the CLEAR address of the GPIO_Px_DOUT register clears the DOUT bits
- Writing the GPIO_Px_DOUTTGL register toggles the corresponding DOUT bits

Reading the GPIO_Px_DOUT register will return its contents. Reading the GPIO_Px_DOUTTGL register will return 0.

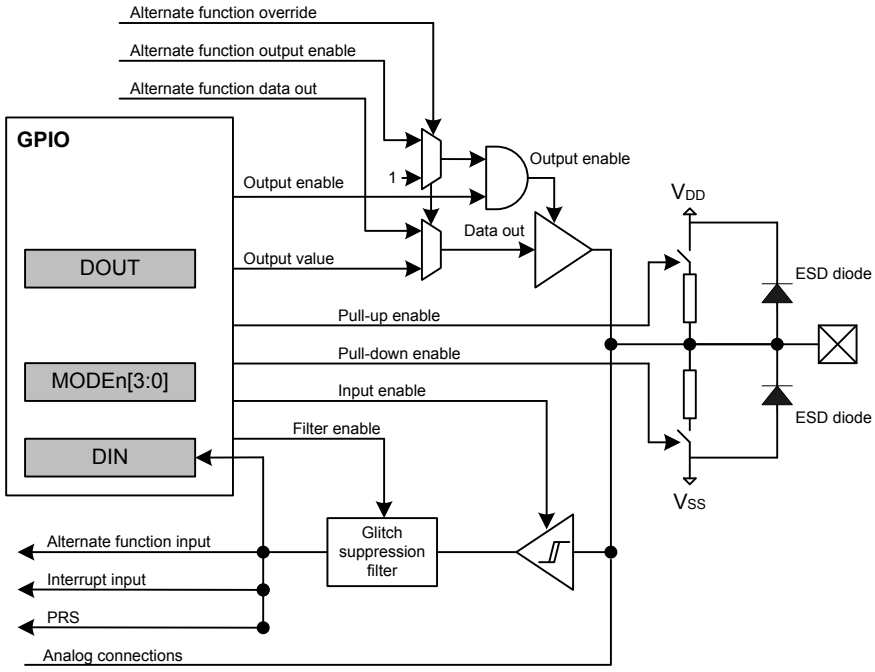


Figure 25.1. Pin Configuration

25.3.1 Pin Configuration

In addition to setting the pins as either outputs or inputs, the GPIO_Px_MODEL and GPIO_Px_MODEH registers can be used for more advanced configurations. GPIO_Px_MODEL contains 8 bit fields named MODEn (n=0,1,..7) which control pins 0-7, while GPIO_Px_MODEH contains 8 bit fields named MODEn (n=8,9,..15) which control pins 8-15. In some modes GPIO_Px_DOUT is also used for extra configurations like pull-up/down and glitch suppression filter enable. Table 25.1 Pin Configuration on page 780 shows the available configurations.

Table 25.1. Pin Configuration

MODEn	Input	Output	DOUT	Pull-down	Pull-up	Alt Port Ctrl	Input Filter	Description
DISABLED	Disabled	Disabled	0					Input disabled
			1		On			Input disabled with pull-up
INPUT	Enabled if not DINDIS		0					Input enabled
			1				On	Input enabled with filter
INPUTPULL			0	On				Input enabled with pull-down
			1		On			Input enabled with pull-up
INPUTPULLFILTER			0	On			On	Input enabled with pull-down and filter
			1		On		On	Input enabled with pull-up and filter
PUSHPULL		Push-pull	x					Push-pull
PUSHPULLALT			x			On		Push-pull with alternate port control values
WIREDOR		Open Source (Wired-OR)	x					Open-source
WIREDORPULLDOWN			x	On				Open-source with pull-down
WIREDAND		Open Drain (Wired-AND)	x					Open-drain
WIREDANDFILTER			x				On	Open-drain with filter
WIREDANDPULLUP			x		On			Open-drain with pull-up
WIREDANDPULLUPFILTER			x		On		On	Open-drain with pull-up and filter
WIREDANDALT			x			On		Open-drain with alternate port control values
WIREDANDALTFILTER			x			On	On	Open-drain with alternate port control values and filter
WIREDANDALTPULLUP			x		On	On		Open-drain with alternate port control values and pull-up
WIREDANDALTPULLUPFILTER			x		On	On	On	Open-drain with alternate port control values, pull-up and filter

MODEn determines which mode the pin is in at a given time. Setting MODEn to DISABLED disables the pin, reducing power consumption to a minimum. When the output driver and input driver are disabled, the pin can be used as a connection for an analog module. An input is enabled by setting MODEn to any value other than DISABLED while DINDIS for the given port is cleared. Set DINDIS to disable

the input of a GPIO port. The pull-up, pull-down and glitch filter function can optionally be applied to the input, see Figure 25.2 Tristated Output with Optional Pull-up or Pull-down on page 781.

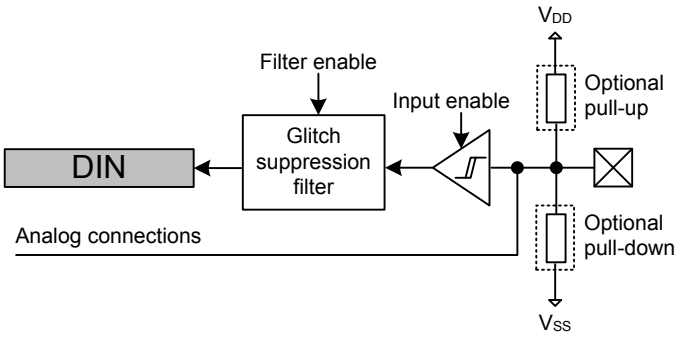


Figure 25.2. Tristated Output with Optional Pull-up or Pull-down

When MODEn is PUSHPULL or PUSHPULLALT, the pin operates in push-pull mode. In this mode, the pin can have alternate port control values and can be driven either high or low, dependent on the value of GPIO_Px_DOUT. The push-pull configuration is shown in Figure 25.3 Push-Pull Configuration on page 781.

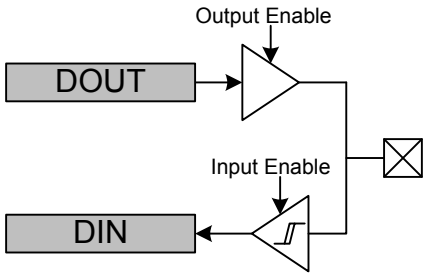


Figure 25.3. Push-Pull Configuration

When MODEn is WIREDOR or WIREDORPULLDOWN, the pin operates in open-source mode (with a pull-down resistor for WIREDORPULLDOWN). When driving a high value in open-source mode, the pull-down is disconnected to save power.

When the mode is prefixed with WIREDAND, the pin operates in open-drain mode as shown in Figure 25.4 Open-drain on page 781. In open-drain mode, the pin can have an input filter, a pull-up, alternate port control values or any combination of these. When driving a low value in open-drain mode, the pull-up is disconnected to save power.

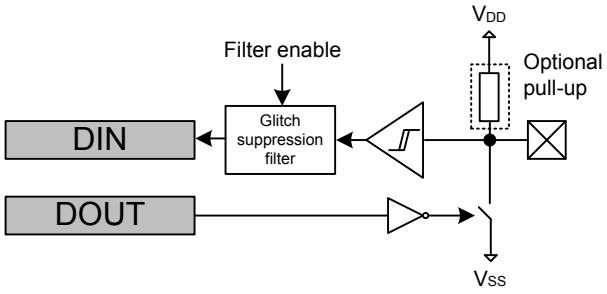


Figure 25.4. Open-drain

25.3.2 Alternate Port Control

The Alternate Port Control allows for additional flexibility of port level settings. A user may setup two different port configurations (normal and alternate modes) and select which is applied on a pin by pin bases. For example you may configure half of port A to use the slowest slew rate while the other half uses a faster slew rate.

Alternate port control is enabled when MODEn is set to any of the ALT enumerated modes (i.e.. PUSHPULLALT). When MODEn is an alternate mode, the pin uses the alternate port control values specified in the DINDISALT and SLEWRATEALT fields in GPIO_Px_CTRL. In all other modes, the port control values are used from the DINDIS and SLEWRATE fields in GPIO_Px_CTRL.

25.3.3 Slew Rate

The slewrate can be applied to pins on a port-by-port basis. The slew rate applied to pins configured using normal MODEn settings can be controlled using the SLEWRATE fields in GPIO_Px_CTRL. The slewrate applied to pins configured using the alternate MODEn settings can be controlled using the SLEWRATEALT field.

The lowest slew rate setting has limited drive strength. That is the current is limited to about 1 mA. This setting provides slow switching and limited drive. A slew rate setting of 1 provides the slowest switching with full drive capability. The maximum recommended setting for most digital I/O is 6. A slew rate setting of 7 should only be used for high-speed clock signals, above 10 MHz. A setting of 7 should not be used on more than one pin per port. Please refer to the datasheet for GPIO rise and fall times.

25.3.4 Input Disable

The pin inputs can be disabled on a port-by-port basis. The input of pins configured using the normal MODEn settings can be disabled by setting DINDIS in GPIO_Px_CTRL. The input of pins configured using the alternate MODEn settings can be disabled by setting DINDISALT.

25.3.5 Configuration Lock

The GPIO configuration registers (GPIO_Px_CTRL, PIO_Px_MODEL, GPIO_xBUSALLOC, GPIO_EXTIPSELL, GPIO_EXTIPINSEL, GPIO_x_yROUTE, and GPIO_xROUTEEN) can be locked by writing any value other than 0xA534 to GPIO_LOCK. Writing the value 0xA534 to the GPIOx_LOCK register unlocks the configuration registers.

25.3.6 EM2 Functionality

While all GPIO pins retain their state in EM2, only pins on port A and B remain fully functional in EM2. Digital peripherals which are active in EM2 must have their resources routed to pins on port A or B to function correctly in EM2. Analog peripherals may use any GPIO pin while in EM2 provided that the ABUS was configured prior to entering EM2. However, analog peripherals that are configured to scan multiple pins while in EM2 (such as the ADC) dynamically reconfigure the ABUS while in EM2 and thus must use only pins on port A and B.

25.3.7 EM4 Functionality

By default GPIO pins revert back to their reset state when EM4 is entered. The GPIO pins can be configured to retain the settings for output enable, output value, pull enable, and pull direction while in EM4.

EM4 GPIO retention is controlled with the EM4IORETMODE field in the EMU_EM4CTRL register:

- Setting EM4IORETMODE to EM4EXIT will cause GPIO retention to persist while in EM4. GPIO state will be reset during wakeup.
- Setting EM4IORETMODE to SWUNLATCH will cause the GPIO retention to persist through EM4 and wakeup, until the EM4UNLATCH bit is written by software. When using SWUNLATCH, the GPIO register values are still reset on wakeup. To ensure the GPIO state does not change, software must re-write the GPIO registers before setting EM4UNLATCH and ending EM4 GPIO retention. Note that the GPIO state cannot be retained through an EM4 wakeup due to a reset (e.g., pin reset or POR reset) - only non-reset methods of EM4 wakeup are supported (e.g., EM4WU IRQ or BURTC IRQ).

See the EMU chapter for additional documentation on EM4IORETMODE and the EM4UNLATCH bit.

25.3.8 EM4 Wakeup

It is possible to trigger a wake-up from EM4 using any of the selectable EM4WU GPIO pins. The wake-up request can be triggered through the pins by enabling the corresponding bit in the GPIO_EM4WUEN register. When EM4 wake-up is enabled for the pin, the input filter is enabled during EM4. This is done to avoid false wake-up caused by glitches. In addition, the polarity of the EM4 wake-up request can be selected using the GPIO_EXTILEVEL register.

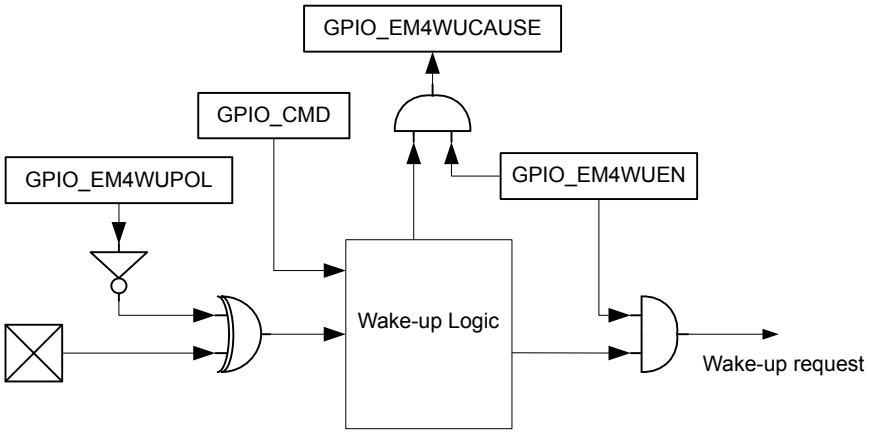


Figure 25.5. EM4 Wake-up Logic

The pins used for EM4 wake-up must be configured as inputs with glitch filters using the GPIO_Px_MODEL register. If the input is disabled and the wakeup polarity is low, a false wakeup will occur when entering EM4. If the input is enabled, the glitch filtered is disabled, and the polarity is set low, a glitch will occur when going into EM4 that will cause an immediate wake-up. Before going down to EM4, it is important to clear the wake-up logic by setting the GPIO_IFC bit, which clears the wake-up logic, including the GPIO_IF register. It is possible to determine which pin caused the EM4WU by reading the GPIO_IF register.

Each EM4WU signal is connected to a fixed pin. Refer to the Alternate Function Table in the device Datasheet for the location of each EM4 wakeup signal.

25.3.9 Debug Connections

25.3.9.1 JTAG Debug Connection

The JTAG Debug Port is a fixed location resource connected directly to specific GPIO pins. Refer to the Alternate Function Table in the device Datasheet for the location of the JTAG signals. By default TMS, TCK, TDO, and TDI pin connections are enabled with internal pull up, pull down, no pull, and pull up resistors, respectively. It is possible to disable these pin connections (and disable the pull resistors) by setting the SWDIOTMSPEN, SWCLKTCKPEN, TDOPEN, and TDIPEN bits in GPIO_DEBUGROUTEPEN to 0.

25.3.9.2 Serial Wire Debug Connection

The SW Debug Port is a fixed location resource connected directly to specific GPIO pins. Refer to the Alternate Function Table in the device Datasheet for the location of the SW Debug port signals. The SWDIO and SWCLK pin connections are enabled by default with internal pull up and pull down resistors, respectively. It is possible to disable these pin connections (and disable the pull resistors) by setting the SWDIOTMSPEN and SWCLKTCKPEN bits in GPIO_DEBUGROUTEPEN to 0.

The Serial Wire Viewer pin, SWV, can be enabled by setting the SWVPEN bit in GPIO_TRACEROUTEPEN.

25.3.9.3 Disabling Debug Connections

When the debug pins are disabled, the device can no longer be accessed by a debugger. A reset will set the debug pins back to their enabled default state. The GPIO_DBGROUTEPEN register can only be updated when the debugger is disconnected from the system. Any attempts to modify GPIO_DBGROUTEPEN when the debugger is connected will not occur. If you do disable the debug pins, make sure you have at least a 3 second timeout at the start of your program code before you disable the debug pins. This way the debugger will have time to connect to the device after a reset and before the pins are disabled.

25.3.9.4 ETM Trace Connections

There is a single trace pin available on the device. One trace clock which can be enabled by setting the TRACECLKPEN bit-field in GPIO_TRACEROUTEPEN. The data pin can be enabled individually by setting TRACEDATA0PEN in GPIO_TRACEROUTEPEN. The trace pins are fixed location resources connected to specific pins. Refer to the Alternate Function Table in the device Datasheet for the location of the SW Debug port signals.

25.3.10 Interrupt Generation

25.3.10.1 Edge Interrupt Generation

The GPIO can generate an interrupt from any edge of the input of any GPIO pin on the device. The edge interrupts have asynchronous sense capability, enabling wake-up from energy modes as low as EM3, see Figure 25.6 Pin n Interrupt Generation on page 784.

Note: In EM2 and EM3, only pins on Port A and Port B are available for edge interrupts. All pins are available for edge interrupts in EM0 and EM1.

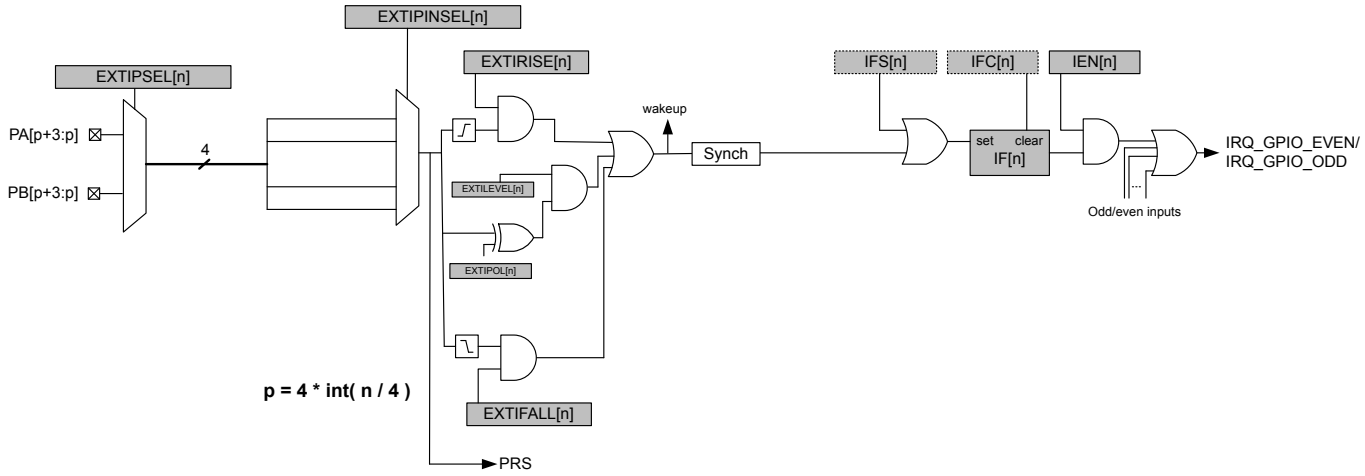


Figure 25.6. Pin n Interrupt Generation

The external pin interrupts are numbered starting with 0. Each interrupt has a corresponding enable bit in the GPIO_IEN register and an interrupt flag bit in the GPIO_IF register. Each interrupt may be used with one of four possible pins on any available port. First select the desired port for each interrupt using the corresponding EXTIPSELx field in the GPIO_EXTIPSELL register. (Some devices with many pins may also have a GPIO_EXTIPSELH register.)

Each interrupt can be mapped to one of four possible pins on the selected port. External interrupts EXTI0 through EXTI3 may be mapped to pins 0,1,2, or 3 on any available port. External interrupts EXTI4 through EXTI7 may be mapped to pins 4,5,6 or 7 on any available port.

Note: Note that while the EXTIEN field in the GPIO_IEN register has 15 bits, the number of useful bits is limited by the number of pins available in the widest port. If the widest port is 8 bits wide, only the first 8 external interrupts are useful.

The selected pin for each interrupt is the base plus the offset. The base for EXTI0 through EXTI3 is 0, while the base for interrupts EXTI4 through EXTI7 is 4. The base may be calculated by taking the interrupt number, dividing by four, then using only the integer portion of the quotient. (BASE = Integer(N/4)

The offset is selected using the corresponding field in the GPIO_EXTIPINSELL register, (Some devices with many pins may also have a GPIO_EXTIPINSELH register.) Subtract the base from the desired pin number to get the offset. For example, to map EXTI5 to pin 7 of PORTA, the base is 4 and the offset will be 3.

The GPIO_EXTIRISE[n] and GPIO_EXTIFALL[n] registers enable sensing of rising and falling edges. By setting the EXT[n] bit in GPIO_IEN, a high interrupt flag n, will trigger one of two interrupt lines. The even interrupt line is triggered by any enabled even numbered interrupt flag index, while the odd interrupt line is triggered by odd flag indexes. The interrupt flags can be set and cleared by software when writing the GPIO_IFS and GPIO_IFC registers. Since the external interrupts are asynchronous, they are sensitive to noise. To increase noise tolerance, the MODEx field(s) in the GPIO_Px_MODEL register, should be set to include glitch filtering for pins that have external interrupts enabled.

25.3.10.2 Level Interrupt Generation on EM4WU Pins

In addition to being an EM4 wake source, any of the EM4WU (EM4 wake-up) pins on the device may be used to generate level-sensitive interrupts in EM0, EM1, EM2, and EM3.

In order to enable the level interrupt, set the EM4WUIENn field in the GPIO_IEN register and the EM4WUENn field in the EM4WUEN register. The EM4WUPOLn field in the GPIO_EM4WUPOL register is used to set the desired polarity for the interrupt.

Upon a level interrupt occurring, the corresponding EM4WU index in the GPIO_IF register will be set along with the odd or even interrupt line depending on the index inside of GPIO_IF. For example, by setting the. EM4WU8 in GPIO_EXTILEVEL and EM4WU[8] in GPIO_IEN, the interrupt flag EM4WU[8] in GPIO_IF will be triggered by a high level on pin EM4WU8 and a interrupt request will be sent on IRQ_GPIO_EVEN.

The wake-up granularity of the level interrupts is based on the settings of the EM4WU field in the GPIO_IEN register and the EM4WUEN field in the GPIO_EM4WUEN register (see Table 25.2 Level Interrupt Energy Mode Wakeup on page 785).

Table 25.2. Level Interrupt Energy Mode Wakeup

EM4WUIENn in GPIO_IEN	EM4WUENn in GPIO_EM4WUEN	Energy Mode Wakeup	Interrupt
x	0	No Wake	No Interrupt
0	1	Wake from EM4	No Interrupt
1	1	Wake from EM1, EM2,EM3, or EM4	Interrupt from EM0, EM1, EM2, or EM3

For example, to configure the device to wake up and generate an interrupt when PD02 (EM4WU9) is logic low:

1. Set bit 9 of EM4WUEN in the GPIO_EM4WUEN register to '1'. This enables the asynchronous wake logic.
2. Set bit 9 of EM4WUIEN in the GPIO_IEN register to '1'. This enables routing of the wake signal to the GPIO_ODD IRQ.
3. Clear bit 9 of EM4WUPOL in the GPIO_EM4WUPOL register to '0'. This indicates that the interrupt should occur when a logic low level is detected at the pin.
4. Enable the GPIO.ODD IRQ. The ODD interrupt is used because the bit index of EM4WUIF in GPIO_IF is odd.

25.3.11 Output to PRS

All pins within a group of four(0-3,4-7,8-11,12-15) from all ports are grouped together to form one PRS producer which outputs to the PRS. The pin from which the output should be taken is selected in the same fashion as the edge interrupts.

PRS output is not affected by the interrupt edge detection logic or gated by the IEN bits. See 25.3.10 Interrupt Generation for an illustration of where the PRS output signal is generated.

25.3.12 Peripheral Resource Routing

Most peripherals have resources that need to be connected to GPIO pins to function. For example, the I2C has SDA and SCL which need to be connected to pins for the I2C to communicate with other ICs. Resources come in three types. Fixed resources are hard-wired to a pin and can only be accessed in that location. For example the LFXO LFXTAL_I and LFXTAL_O resources are only available on one pin each. Digital route-able resources are connected to pins through the 25.3.12.1 Digital Bus (DBUS) which allows for extremely flexible resource placement. Analog route-able resources are connected to pins though the 25.3.12.2 Analog Bus (ABUS) which provides extremely flexible resource placement.

The locations of fixed resources and the limitations of ABUS and DBUS on each device can be found in the device data sheet.

25.3.12.1 Digital Bus (DBUS)

The Digital Bus (DBUS) is an any-to-any switch matrix between peripheral resources and GPIO pins as shown in Figure 25.7 Digital Bus Interconnect on page 786. There are two DBUSes on the EFR32xG22 - one serving ports A and B and the other ports C and D. Not all peripherals have access to both DBUSes.

To connect a resource to a pin, first select the desired PORT and PIN in the GPIO_x_yROUTE register, where x is the peripheral name and y is the resource name. Once the pin is selected, the resource must be enabled by setting its enable bit in the appropriate GPIO_x_ROUTEEN register. For example, to route the TX resource of USART1 to PB3, set PORT to 0x1 and PIN to 0x3 in GPIO_USART1_TXROUTE. Then set the GPIO_USART1_ROUTEN.TXEN bit.

Any pin connected to a digital resource should be properly configured for that resource. For example USART1 TX should be configured as push-pull and USART1 RX should be configured as an input.

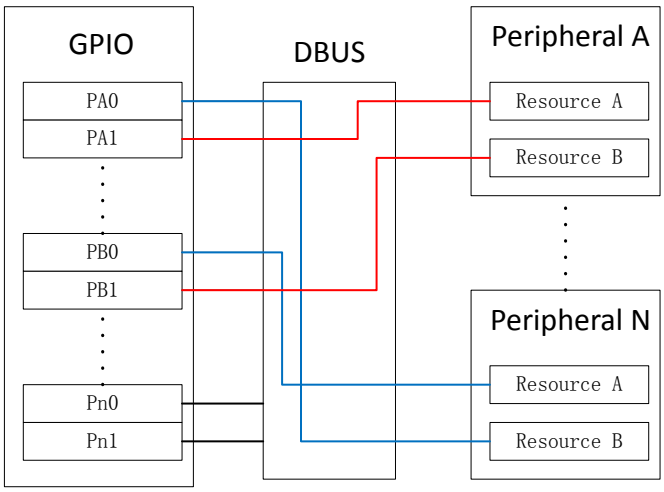


Figure 25.7. Digital Bus Interconnect

25.3.12.2 Analog Bus (ABUS)

Analog peripherals may be connected to any pins on port A, B, C, or D via the Analog Bus. There are three analog buses on the EFR32xG22: one dedicated to Port A (ABUSA), one dedicated to port B (ABUSB), and one that serves both ports C and D (ABUSCD). The specific pin and port selection for analog resources are configured in the analog peripherals. Refer to the respective analog peripheral chapter for this information. However, the GPIO block must be configured to grant the peripheral access to an ABUS before any connection can be made.

Up to two analog peripherals may be given access to an ABUS at any one time and the even/odd pins of each bus are configured independently. This means that a single bus may have up to four different analog peripherals connected to it: two on the even pins and two on the odd pins. The GPIO_ABUSxALLOC register, where x is the port, determines which peripherals have access to the bus. To grant a peripheral access to the bus even pins select it in either the EVEN0 or EVEN1 field. To grant a peripheral access to the bus odd pins select it in either the ODD0 or ODD1 fields.

When a differential connection is being used, positive inputs are restricted to the EVEN pins and negative inputs are restricted to the ODD pins. When a single ended connection is being used, the positive input is avaliable on all pins.

Peripherals may be given access to as many buses as desired. For example the ADC may be given access to ABUSA, ABUSB, and ABUSCD allowing it to select any pin on ports A-D. If two peripherals select the same port and pin the ABUS will make both connections simultaneously, effectively connecting the two peripherals together.

Any pin connected to an analog resource should be configured to input DISABLED as described in 25.3.1 Pin Configuration

The process for configuring an analog peripheral to access a pin through the ABUS is as follows:

- Configure the desired analog port pins to input DISABLED mode in the corresponding GPIO_PORTx_MODEL/H register.
- Configure the corresponding GPIO_xBUSALLOC field to grant access to the desired peripheral on the desired ABUS.
- Configure the analog peripheral to select the desired port and channel as described in the peripheral chapter.

25.3.12.3 Pin Function Tables

This section details the functions and GPIO pins available on the most fully-featured devices in the EFR32xG22 family. Availability of GPIO may vary on smaller packages. Refer to the device datasheet for specific peripheral and GPIO availability. Fixed-pin peripheral resources are shown in Table 25.3 GPIO Fixed Pin Function Table on page 787, ABUS routing options are listed in Table 25.4 ABUS Routing Table on page 787, and DBUS routing options are listed in Table 25.5 DBUS Routing Table on page 787

Table 25.3. GPIO Fixed Pin Function Table

GPIO	Alternate Function				
PC00	GPIO.EM4WU6	GPIO.THMSW_EN			
PC05	GPIO.EM4WU7				
PC07	GPIO.EM4WU8				
PB03	GPIO.EM4WU4				
PB01	GPIO.EM4WU3				
PB00	IADC0.VREFN				
PA00	IADC0.VREFP				
PA01	GPIO.SWCLK				
PA02	GPIO.SWDIO				
PA03	GPIO.SWV	GPIO.TDO	GPIO.TRACEDA-TA0		
PA04	GPIO.TDI	GPIO.TRACECLK			
PA05	GPIO.EM4WU0				
PD02	GPIO.EM4WU9				
PD01	LFXO.LFXTAL_I	LFXO.LF_EXTCLK			
PD00	LFXO.LFXTAL_O				

Table 25.4. ABUS Routing Table

Peripheral	Signal	PA		PB		PC		PD	
		EVEN	ODD	EVEN	ODD	EVEN	ODD	EVEN	ODD
IADC0	ana_neg	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	ana_pos	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 25.5. DBUS Routing Table

Peripheral.Resource	PORT			
	PA	PB	PC	PD
CMU.CLKIN0			Available	Available
CMU.CLKOUT0			Available	Available
CMU.CLKOUT1			Available	Available
CMU.CLKOUT2	Available	Available		

Peripheral.Resource	PORT			
	PA	PB	PC	PD
EUART0.CTS	Available	Available	Available	Available
EUART0.RTS	Available	Available	Available	Available
EUART0.RX	Available	Available	Available	Available
EUART0.TX	Available	Available	Available	Available
FRC.DCLK			Available	Available
FRC.DFRAME			Available	Available
FRC.DOUT			Available	Available
I2C0.SCL	Available	Available	Available	Available
I2C0.SDA	Available	Available	Available	Available
I2C1.SCL			Available	Available
I2C1.SDA			Available	Available
LETIMER0.OUT0	Available	Available		
LETIMER0.OUT1	Available	Available		
MODEM.ANT0	Available	Available	Available	Available
MODEM.ANT1	Available	Available	Available	Available
MODEM.ANT_ROLL_OVER			Available	Available
MODEM.ANT_RR0			Available	Available
MODEM.ANT_RR1			Available	Available
MODEM.ANT_RR2			Available	Available
MODEM.ANT_RR3			Available	Available
MODEM.ANT_RR4			Available	Available
MODEM.ANT_RR5			Available	Available
MODEM.ANT_SW_EN			Available	Available
MODEM.ANT_SW_US			Available	Available
MODEM.ANT_TRIG			Available	Available
MODEM.ANT_TRIG_STOP			Available	Available
MODEM.DCLK	Available	Available		
MODEM.DIN	Available	Available		
MODEM.DOUT	Available	Available		
PDM.CLK	Available	Available	Available	Available
PDM.DAT0	Available	Available	Available	Available
PDM.DAT1	Available	Available	Available	Available
PRS.ASYNCH0	Available	Available		
PRS.ASYNCH1	Available	Available		
PRS.ASYNCH10			Available	Available
PRS.ASYNCH11			Available	Available

Peripheral.Resource	PORT			
	PA	PB	PC	PD
PRS.ASYNCH2	Available	Available		
PRS.ASYNCH3	Available	Available		
PRS.ASYNCH4	Available	Available		
PRS.ASYNCH5	Available	Available		
PRS.ASYNCH6			Available	Available
PRS.ASYNCH7			Available	Available
PRS.ASYNCH8			Available	Available
PRS.ASYNCH9			Available	Available
PRS.SYNCH0	Available	Available	Available	Available
PRS.SYNCH1	Available	Available	Available	Available
PRS.SYNCH2	Available	Available	Available	Available
PRS.SYNCH3	Available	Available	Available	Available
TIMER0.CC0	Available	Available	Available	Available
TIMER0.CC1	Available	Available	Available	Available
TIMER0.CC2	Available	Available	Available	Available
TIMER0.CDTI0	Available	Available	Available	Available
TIMER0.CDTI1	Available	Available	Available	Available
TIMER0.CDTI2	Available	Available	Available	Available
TIMER1.CC0	Available	Available	Available	Available
TIMER1.CC1	Available	Available	Available	Available
TIMER1.CC2	Available	Available	Available	Available
TIMER1.CDTI0	Available	Available	Available	Available
TIMER1.CDTI1	Available	Available	Available	Available
TIMER1.CDTI2	Available	Available	Available	Available
TIMER2.CC0	Available	Available		
TIMER2.CC1	Available	Available		
TIMER2.CC2	Available	Available		
TIMER2.CDTI0	Available	Available		
TIMER2.CDTI1	Available	Available		
TIMER2.CDTI2	Available	Available		
TIMER3.CC0			Available	Available
TIMER3.CC1			Available	Available
TIMER3.CC2			Available	Available
TIMER3.CDTI0			Available	Available
TIMER3.CDTI1			Available	Available
TIMER3.CDTI2			Available	Available

Peripheral.Resource	PORT			
	PA	PB	PC	PD
TIMER4.CC0	Available	Available		
TIMER4.CC1	Available	Available		
TIMER4.CC2	Available	Available		
TIMER4.CDTI0	Available	Available		
TIMER4.CDTI1	Available	Available		
TIMER4.CDTI2	Available	Available		
USART0.CLK	Available	Available	Available	Available
USART0.CS	Available	Available	Available	Available
USART0.CTS	Available	Available	Available	Available
USART0.RTS	Available	Available	Available	Available
USART0.RX	Available	Available	Available	Available
USART0.TX	Available	Available	Available	Available
USART1.CLK	Available	Available		
USART1.CS	Available	Available		
USART1.CTS	Available	Available		
USART1.RTS	Available	Available		
USART1.RX	Available	Available		
USART1.TX	Available	Available		

25.4 Synchronization

To avoid metastability in synchronous logic connected to the pins, all inputs are synchronized with double flip-flops. The flip-flops for the input data run on these HFBUSCLK. Consequently, when a pin changes state, the change will propagate to GPIO_Px_DIN after two 2 HFBUSCLK cycles. Synchronization (also running on the HFBUSCLK) is also added for interrupt input. To save power when the external interrupts or level interrupts are not used, the synchronization flip-flops for these can be turned off by clearing the EXTINT field in the GPIO_IEN register.

25.5 GPIO Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	GPIO_PORTA_CTRL	RW	Port control
0x004	GPIO_PORTA_MODEL	RW	mode low
0x00C	GPIO_PORTA_MODEH	RW	mode high
0x010	GPIO_PORTA_DOUT	RW	data out
0x014	GPIO_PORTA_DIN	RH	data in
0x030	GPIO_PORTB_CTRL	RW	Port control
0x034	GPIO_PORTB_MODEL	RW	mode low
0x040	GPIO_PORTB_DOUT	RW	data out
0x044	GPIO_PORTB_DIN	RH	data in
0x060	GPIO_PORTC_CTRL	RW	Port control
0x064	GPIO_PORTC_MODEL	RW	mode low
0x070	GPIO_PORTC_DOUT	RW	data out
0x074	GPIO_PORTC_DIN	RH	data in
0x090	GPIO_PORTD_CTRL	RW	Port control
0x094	GPIO_PORTD_MODEL	RW	mode low
0x0A0	GPIO_PORTD_DOUT	RW	data out
0x0A4	GPIO_PORTD_DIN	RH	data in
0x300	GPIO_LOCK	W	Lock Register
0x310	GPIO_GPIOLOCKSTATUS	RH	Lock Status
0x320	GPIO_ABUSALLOC	RW	A Bus allocation
0x324	GPIO_BBUSALLOC	RW	B Bus allocation
0x328	GPIO_CDBUSALLOC	RW	CD Bus allocation
0x400	GPIO_EXTIPSELL	RW	External Interrupt Port Select Low
0x404	GPIO_EXTIPSELH	RW	External interrupt Port Select High
0x408	GPIO_EXTIPINSELL	RW	External Interrupt Pin Select Low
0x40C	GPIO_EXTIPINSELH	RW	External Interrupt Pin Select High
0x410	GPIO_EXTIRISE	RW	External Interrupt Rising Edge Trigger
0x414	GPIO_EXTIFALL	RW	External Interrupt Falling Edge Trigger
0x420	GPIO_IF	RWH INTFLAG	Interrupt Flag
0x424	GPIO_IEN	RW	Interrupt Enable
0x42C	GPIO_EM4WUEN	RW	EM4 wakeup enable
0x430	GPIO_EM4WUPOL	RW	EM4 wakeup polarity
0x440	GPIO_DBGROUTEPEN	RW	Debugger Route Pin enable
0x444	GPIO_TRACEROUTEPEN	RW	Trace Route Pin Enable
0x450	GPIO_CMU_ROUTEEN	RW	CMU pin enable

Offset	Name	Type	Description
0x454	GPIO_CMU_CLKIN0ROUTE	RW	CLKIN0 port/pin select
0x458	GPIO_CMU_CLKOUT0ROUTE	RW	CLKOUT0 port/pin select
0x45C	GPIO_CMU_CLKOUT1ROUTE	RW	CLKOUT1 port/pin select
0x460	GPIO_CMU_CLKOUT2ROUTE	RW	CLKOUT2 port/pin select
0x46C	GPIO_DCDC_ROUTEEN	RW	DCDC pin enable
0x47C	GPIO_FRC_ROUTEEN	RW	FRC pin enable
0x480	GPIO_FRC_DCLKROUTE	RW	DCLK port/pin select
0x484	GPIO_FRC_DFRAMEROUTE	RW	DFRAME port/pin select
0x488	GPIO_FRC_DOUTROUTE	RW	DOUT port/pin select
0x490	GPIO_I2C0_ROUTEEN	RW	I2C0 pin enable
0x494	GPIO_I2C0_SCLROUTE	RW	SCL port/pin select
0x498	GPIO_I2C0_SDAROUTE	RW	SDA port/pin select
0x4A0	GPIO_I2C1_ROUTEEN	RW	I2C1 pin enable
0x4A4	GPIO_I2C1_SCLROUTE	RW	SCL port/pin select
0x4A8	GPIO_I2C1_SDAROUTE	RW	SDA port/pin select
0x4B0	GPIO_LETIMER0_ROUTEEN	RW	LETIMER pin enable
0x4B4	GPIO_LETIMER0_OUT0ROUTE	RW	OUT0 port/pin select
0x4B8	GPIO_LETIMER0_OUT1ROUTE	RW	OUT1 port/pin select
0x4C0	GPIO_EUART0_ROUTEEN	RW	EUART pin enable
0x4C4	GPIO_EUART0_CTSROUTE	RW	CTS port/pin select
0x4C8	GPIO_EUART0_RTSROUTE	RW	RTS port/pin select
0x4CC	GPIO_EUART0_RXROUTE	RW	RX port/pin select
0x4D0	GPIO_EUART0_TXROUTE	RW	TX port/pin select
0x4D8	GPIO_MODEM_ROUTEEN	RW	MODEM pin enable
0x4DC	GPIO_MODEM_ANT0ROUTE	RW	ANT0 port/pin select
0x4E0	GPIO_MODEM_ANT1ROUTE	RW	ANT1 port/pin select
0x4E4	GPIO_MODEM_ANTROLLO-VERROUTE	RW	ANTROLLOVER port/pin select
0x4E8	GPIO_MO-DEM_ANTRR0ROUTE	RW	ANTRR0 port/pin select
0x4EC	GPIO_MO-DEM_ANTRR1ROUTE	RW	ANTRR1 port/pin select
0x4F0	GPIO_MO-DEM_ANTRR2ROUTE	RW	ANTRR2 port/pin select
0x4F4	GPIO_MO-DEM_ANTRR3ROUTE	RW	ANTRR3 port/pin select
0x4F8	GPIO_MO-DEM_ANTRR4ROUTE	RW	ANTRR4 port/pin select
0x4FC	GPIO_MO-DEM_ANTRR5ROUTE	RW	ANTRR5 port/pin select

Offset	Name	Type	Description
0x500	GPIO_MODEM_ANTSWEN-ROUTE	RW	ANTSWEN port/pin select
0x504	GPIO_MODEM_ANTSWUS-ROUTE	RW	ANTSWUS port/pin select
0x508	GPIO_MODEM_ANTTRIG-ROUTE	RW	ANTTRIG port/pin select
0x50C	GPIO_MODEM_ANTTRIGSTOP-ROUTE	RW	ANTTRIGSTOP port/pin select
0x510	GPIO_MODEM_DCLKROUTE	RW	DCLK port/pin select
0x514	GPIO_MODEM_DINROUTE	RW	DIN port/pin select
0x518	GPIO_MODEM_DOUTROUTE	RW	DOUT port/pin select
0x520	GPIO_PDM_ROUTEEN	RW	PDM pin enable
0x524	GPIO_PDM_CLKROUTE	RW	CLK port/pin select
0x528	GPIO_PDM_DAT0ROUTE	RW	DAT0 port/pin select
0x52C	GPIO_PDM_DAT1ROUTE	RW	DAT1 port/pin select
0x534	GPIO_PRS0_ROUTEEN	RW	PRS0 pin enable
0x538	GPIO_PRS0_ASYNCH0ROUTE	RW	ASYNCH0 port/pin select
0x53C	GPIO_PRS0_ASYNCH1ROUTE	RW	ASYNCH1 port/pin select
0x540	GPIO_PRS0_ASYNCH2ROUTE	RW	ASYNCH2 port/pin select
0x544	GPIO_PRS0_ASYNCH3ROUTE	RW	ASYNCH3 port/pin select
0x548	GPIO_PRS0_ASYNCH4ROUTE	RW	ASYNCH4 port/pin select
0x54C	GPIO_PRS0_ASYNCH5ROUTE	RW	ASYNCH5 port/pin select
0x550	GPIO_PRS0_ASYNCH6ROUTE	RW	ASYNCH6 port/pin select
0x554	GPIO_PRS0_ASYNCH7ROUTE	RW	ASYNCH7 port/pin select
0x558	GPIO_PRS0_ASYNCH8ROUTE	RW	ASYNCH8 port/pin select
0x55C	GPIO_PRS0_ASYNCH9ROUTE	RW	ASYNCH9 port/pin select
0x560	GPIO_PRS0_ASYNCH10ROUTE	RW	ASYNCH10 port/pin select
0x564	GPIO_PRS0_ASYNCH11ROUTE	RW	ASYNCH11 port/pin select
0x568	GPIO_PRS0_SYNCH0ROUTE	RW	SYNCH0 port/pin select
0x56C	GPIO_PRS0_SYNCH1ROUTE	RW	SYNCH1 port/pin select
0x570	GPIO_PRS0_SYNCH2ROUTE	RW	SYNCH2 port/pin select
0x574	GPIO_PRS0_SYNCH3ROUTE	RW	SYNCH3 port/pin select
0x57C	GPIO_TIMER0_ROUTEEN	RW	TIMER0 pin enable
0x580	GPIO_TIMER0_CC0ROUTE	RW	CC0 port/pin select
0x584	GPIO_TIMER0_CC1ROUTE	RW	CC1 port/pin select
0x588	GPIO_TIMER0_CC2ROUTE	RW	CC2 port/pin select
0x58C	GPIO_TIMER0_CCC0ROUTE	RW	CCC0 port/pin select

Offset	Name	Type	Description
0x590	GPIO_TIMER0_CCC1ROUTE	RW	CCC1 port/pin select
0x594	GPIO_TIMER0_CCC2ROUTE	RW	CCC2 port/pin select
0x59C	GPIO_TIMER1_ROUTEEN	RW	TIMER1 pin enable
0x5A0	GPIO_TIMER1_CC0ROUTE	RW	CC0 port/pin select
0x5A4	GPIO_TIMER1_CC1ROUTE	RW	CC1 port/pin select
0x5A8	GPIO_TIMER1_CC2ROUTE	RW	CC2 port/pin select
0x5AC	GPIO_TIMER1_CCC0ROUTE	RW	CCC0 port/pin select
0x5B0	GPIO_TIMER1_CCC1ROUTE	RW	CCC1 port/pin select
0x5B4	GPIO_TIMER1_CCC2ROUTE	RW	CCC2 port/pin select
0x5BC	GPIO_TIMER2_ROUTEEN	RW	TIMER2 pin enable
0x5C0	GPIO_TIMER2_CC0ROUTE	RW	CC0 port/pin select
0x5C4	GPIO_TIMER2_CC1ROUTE	RW	CC1 port/pin select
0x5C8	GPIO_TIMER2_CC2ROUTE	RW	CC2 port/pin select
0x5CC	GPIO_TIMER2_CCC0ROUTE	RW	CCC0 port/pin select
0x5D0	GPIO_TIMER2_CCC1ROUTE	RW	CCC1 port/pin select
0x5D4	GPIO_TIMER2_CCC2ROUTE	RW	CCC2 port/pin select
0x5DC	GPIO_TIMER3_ROUTEEN	RW	TIMER3 pin enable
0x5E0	GPIO_TIMER3_CC0ROUTE	RW	CC0 port/pin select
0x5E4	GPIO_TIMER3_CC1ROUTE	RW	CC1 port/pin select
0x5E8	GPIO_TIMER3_CC2ROUTE	RW	CC2 port/pin select
0x5EC	GPIO_TIMER3_CCC0ROUTE	RW	CCC0 port/pin select
0x5F0	GPIO_TIMER3_CCC1ROUTE	RW	CCC1 port/pin select
0x5F4	GPIO_TIMER3_CCC2ROUTE	RW	CCC2 port/pin select
0x5FC	GPIO_TIMER4_ROUTEEN	RW	TIMER4 pin enable
0x600	GPIO_TIMER4_CC0ROUTE	RW	CC0 port/pin select
0x604	GPIO_TIMER4_CC1ROUTE	RW	CC1 port/pin select
0x608	GPIO_TIMER4_CC2ROUTE	RW	CC2 port/pin select
0x60C	GPIO_TIMER4_CCC0ROUTE	RW	CCC0 port/pin select
0x610	GPIO_TIMER4_CCC1ROUTE	RW	CCC1 port/pin select
0x614	GPIO_TIMER4_CCC2ROUTE	RW	CCC2 port/pin select
0x61C	GPIO_USART0_ROUTEEN	RW	USART0 pin enable
0x620	GPIO_USART0_CSROUTE	RW	CS port/pin select
0x624	GPIO_USART0_CTSROUTE	RW	CTS port/pin select
0x628	GPIO_USART0_RTSROUTE	RW	RTS port/pin select
0x62C	GPIO_USART0_RXROUTE	RW	RX port/pin select
0x630	GPIO_USART0_CLKROUTE	RW	SCLK port/pin select
0x634	GPIO_USART0_TXROUTE	RW	TX port/pin select

Offset	Name	Type	Description
0x63C	GPIO_USART1_ROUTEEN	RW	USART1 pin enable
0x640	GPIO_USART1_CSROUTE	RW	CS port/pin select
0x644	GPIO_USART1_CTSROUTE	RW	CTS port/pin select
0x648	GPIO_USART1_RTSROUTE	RW	RTS port/pin select
0x64C	GPIO_USART1_RXROUTE	RW	RX port/pin select
0x650	GPIO_USART1_CLKROUTE	RW	SCLK port/pin select
0x654	GPIO_USART1_TXROUTE	RW	TX port/pin select
0x1000	GPIO_PORTA_CTRL_SET	RW	Port control
0x1004	GPIO_PORTA_MODEL_SET	RW	mode low
0x100C	GPIO_PORTA_MODEH_SET	RW	mode high
0x1010	GPIO_PORTA_DOUT_SET	RW	data out
0x1014	GPIO_PORTA_DIN_SET	RH	data in
0x1030	GPIO_PORTB_CTRL_SET	RW	Port control
0x1034	GPIO_PORTB_MODEL_SET	RW	mode low
0x1040	GPIO_PORTB_DOUT_SET	RW	data out
0x1044	GPIO_PORTB_DIN_SET	RH	data in
0x1060	GPIO_PORTC_CTRL_SET	RW	Port control
0x1064	GPIO_PORTC_MODEL_SET	RW	mode low
0x1070	GPIO_PORTC_DOUT_SET	RW	data out
0x1074	GPIO_PORTC_DIN_SET	RH	data in
0x1090	GPIO_PORTD_CTRL_SET	RW	Port control
0x1094	GPIO_PORTD_MODEL_SET	RW	mode low
0x10A0	GPIO_PORTD_DOUT_SET	RW	data out
0x10A4	GPIO_PORTD_DIN_SET	RH	data in
0x1300	GPIO_LOCK_SET	W	Lock Register
0x1310	GPIO_GPIOLOCKSTATUS_SET	RH	Lock Status
0x1320	GPIO_ABUSALLOC_SET	RW	A Bus allocation
0x1324	GPIO_BBUSALLOC_SET	RW	B Bus allocation
0x1328	GPIO_CDBUSALLOC_SET	RW	CD Bus allocation
0x1400	GPIO_EXTIPSELL_SET	RW	External Interrupt Port Select Low
0x1404	GPIO_EXTIPSELH_SET	RW	External interrupt Port Select High
0x1408	GPIO_EXTIPINSELL_SET	RW	External Interrupt Pin Select Low
0x140C	GPIO_EXTIPINSELH_SET	RW	External Interrupt Pin Select High
0x1410	GPIO_EXTIRISE_SET	RW	External Interrupt Rising Edge Trigger
0x1414	GPIO_EXTIFALL_SET	RW	External Interrupt Falling Edge Trigger
0x1420	GPIO_IF_SET	RWH INTFLAG	Interrupt Flag
0x1424	GPIO_IEN_SET	RW	Interrupt Enable

Offset	Name	Type	Description
0x142C	GPIO_EM4WUEN_SET	RW	EM4 wakeup enable
0x1430	GPIO_EM4WUPOL_SET	RW	EM4 wakeup polarity
0x1440	GPIO_DBGROUTEPEN_SET	RW	Debugger Route Pin enable
0x1444	GPIO_TRACEROUTEPEN_SET	RW	Trace Route Pin Enable
0x1450	GPIO_CMU_ROUTEEN_SET	RW	CMU pin enable
0x1454	GPIO_CMU_CLKIN0ROUTE_SET	RW	CLKIN0 port/pin select
0x1458	GPIO_CMU_CLKOUT0ROUTE_SET	RW	CLKOUT0 port/pin select
0x145C	GPIO_CMU_CLKOUT1ROUTE_SET	RW	CLKOUT1 port/pin select
0x1460	GPIO_CMU_CLKOUT2ROUTE_SET	RW	CLKOUT2 port/pin select
0x146C	GPIO_DCDC_ROUTEEN_SET	RW	DCDC pin enable
0x147C	GPIO_FRC_ROUTEEN_SET	RW	FRC pin enable
0x1480	GPIO_FRC_DCLKROUTE_SET	RW	DCLK port/pin select
0x1484	GPIO_FRC_DFRAME-ROUTE_SET	RW	DFRAME port/pin select
0x1488	GPIO_FRC_DOUTROUTE_SET	RW	DOUT port/pin select
0x1490	GPIO_I2C0_ROUTEEN_SET	RW	I2C0 pin enable
0x1494	GPIO_I2C0_SCLROUTE_SET	RW	SCL port/pin select
0x1498	GPIO_I2C0_SDAROUTE_SET	RW	SDA port/pin select
0x14A0	GPIO_I2C1_ROUTEEN_SET	RW	I2C1 pin enable
0x14A4	GPIO_I2C1_SCLROUTE_SET	RW	SCL port/pin select
0x14A8	GPIO_I2C1_SDAROUTE_SET	RW	SDA port/pin select
0x14B0	GPIO_LETIMER0_ROU-TEEN_SET	RW	LETIMER pin enable
0x14B4	GPIO_LETIM-ER0_OUT0ROUTE_SET	RW	OUT0 port/pin select
0x14B8	GPIO_LETIM-ER0_OUT1ROUTE_SET	RW	OUT1 port/pin select
0x14C0	GPIO_EUART0_ROU-TEEN_SET	RW	EUART pin enable
0x14C4	GPIO_EU-ART0_CTSROUTE_SET	RW	CTS port/pin select
0x14C8	GPIO_EU-ART0_RTSROUTE_SET	RW	RTS port/pin select
0x14CC	GPIO_EU-ART0_RXROUTE_SET	RW	RX port/pin select
0x14D0	GPIO_EUART0_TXROUTE_SET	RW	TX port/pin select
0x14D8	GPIO_MODEM_ROUTEEN_SET	RW	MODEM pin enable

Offset	Name	Type	Description
0x14DC	GPIO_MODEM_ANT0ROUTE_SET	RW	ANT0 port/pin select
0x14E0	GPIO_MODEM_ANT1ROUTE_SET	RW	ANT1 port/pin select
0x14E4	GPIO_MODEM_ANTROLLOVERROUTE_SET	RW	ANTROLLOVER port/pin select
0x14E8	GPIO_MODEM_ANTRR0ROUTE_SET	RW	ANTRR0 port/pin select
0x14EC	GPIO_MODEM_ANTRR1ROUTE_SET	RW	ANTRR1 port/pin select
0x14F0	GPIO_MODEM_ANTRR2ROUTE_SET	RW	ANTRR2 port/pin select
0x14F4	GPIO_MODEM_ANTRR3ROUTE_SET	RW	ANTRR3 port/pin select
0x14F8	GPIO_MODEM_ANTRR4ROUTE_SET	RW	ANTRR4 port/pin select
0x14FC	GPIO_MODEM_ANTRR5ROUTE_SET	RW	ANTRR5 port/pin select
0x1500	GPIO_MODEM_ANTSWENROUTE_SET	RW	ANTSWEN port/pin select
0x1504	GPIO_MODEM_ANTSWUSROUTE_SET	RW	ANTSWUS port/pin select
0x1508	GPIO_MODEM_ANTTRIGROUTE_SET	RW	ANTTRIG port/pin select
0x150C	GPIO_MODEM_ANTTRIGSTOPROUTE_SET	RW	ANTTRIGSTOP port/pin select
0x1510	GPIO_MODEM_DCLKROUTE_SET	RW	DCLK port/pin select
0x1514	GPIO_MODEM_DINROUTE_SET	RW	DIN port/pin select
0x1518	GPIO_MODEM_DOUTROUTE_SET	RW	DOUT port/pin select
0x1520	GPIO_PDM_ROUTEEN_SET	RW	PDM pin enable
0x1524	GPIO_PDM_CLKROUTE_SET	RW	CLK port/pin select
0x1528	GPIO_PDM_DAT0ROUTE_SET	RW	DAT0 port/pin select
0x152C	GPIO_PDM_DAT1ROUTE_SET	RW	DAT1 port/pin select
0x1534	GPIO_PRS0_ROUTEEN_SET	RW	PRS0 pin enable
0x1538	GPIO_PRS0_ASYNCH0ROUTE_SET	RW	ASYNCH0 port/pin select
0x153C	GPIO_PRS0_ASYNCH1ROUTE_SET	RW	ASYNCH1 port/pin select
0x1540	GPIO_PRS0_ASYNCH2ROUTE_SET	RW	ASYNCH2 port/pin select
0x1544	GPIO_PRS0_ASYNCH3ROUTE_SET	RW	ASYNCH3 port/pin select

Offset	Name	Type	Description
0x1548	GPIO_PRS0_ASYNCH4ROUTE_SET	RW	ASYNCH4 port/pin select
0x154C	GPIO_PRS0_ASYNCH5ROUTE_SET	RW	ASYNCH5 port/pin select
0x1550	GPIO_PRS0_ASYNCH6ROUTE_SET	RW	ASYNCH6 port/pin select
0x1554	GPIO_PRS0_ASYNCH7ROUTE_SET	RW	ASYNCH7 port/pin select
0x1558	GPIO_PRS0_ASYNCH8ROUTE_SET	RW	ASYNCH8 port/pin select
0x155C	GPIO_PRS0_ASYNCH9ROUTE_SET	RW	ASYNCH9 port/pin select
0x1560	GPIO_PRS0_ASYNCH10ROUTE_SET	RW	ASYNCH10 port/pin select
0x1564	GPIO_PRS0_ASYNCH11ROUTE_SET	RW	ASYNCH11 port/pin select
0x1568	GPIO_PRS0_SYNCH0ROUTE_SET	RW	SYNCH0 port/pin select
0x156C	GPIO_PRS0_SYNCH1ROUTE_SET	RW	SYNCH1 port/pin select
0x1570	GPIO_PRS0_SYNCH2ROUTE_SET	RW	SYNCH2 port/pin select
0x1574	GPIO_PRS0_SYNCH3ROUTE_SET	RW	SYNCH3 port/pin select
0x157C	GPIO_TIMER0_ROUTEEN_SET	RW	TIMER0 pin enable
0x1580	GPIO_TIMER0_CC0ROUTE_SET	RW	CC0 port/pin select
0x1584	GPIO_TIMER0_CC1ROUTE_SET	RW	CC1 port/pin select
0x1588	GPIO_TIMER0_CC2ROUTE_SET	RW	CC2 port/pin select
0x158C	GPIO_TIMER0_CCC0ROUTE_SET	RW	CCC0 port/pin select
0x1590	GPIO_TIMER0_CCC1ROUTE_SET	RW	CCC1 port/pin select
0x1594	GPIO_TIMER0_CCC2ROUTE_SET	RW	CCC2 port/pin select
0x159C	GPIO_TIMER1_ROUTEEN_SET	RW	TIMER1 pin enable
0x15A0	GPIO_TIMER1_CC0ROUTE_SET	RW	CC0 port/pin select
0x15A4	GPIO_TIMER1_CC1ROUTE_SET	RW	CC1 port/pin select
0x15A8	GPIO_TIMER1_CC2ROUTE_SET	RW	CC2 port/pin select
0x15AC	GPIO_TIMER1_CCC0ROUTE_SET	RW	CCC0 port/pin select

Offset	Name	Type	Description
0x15B0	GPIO_TIMER1_CCC1ROUTE_SET	RW	CCC1 port/pin select
0x15B4	GPIO_TIMER1_CCC2ROUTE_SET	RW	CCC2 port/pin select
0x15BC	GPIO_TIMER2_ROUTEEN_SET	RW	TIMER2 pin enable
0x15C0	GPIO_TIMER2_CC0ROUTE_SET	RW	CC0 port/pin select
0x15C4	GPIO_TIMER2_CC1ROUTE_SET	RW	CC1 port/pin select
0x15C8	GPIO_TIMER2_CC2ROUTE_SET	RW	CC2 port/pin select
0x15CC	GPIO_TIMER2_CCC0ROUTE_SET	RW	CCC0 port/pin select
0x15D0	GPIO_TIMER2_CCC1ROUTE_SET	RW	CCC1 port/pin select
0x15D4	GPIO_TIMER2_CCC2ROUTE_SET	RW	CCC2 port/pin select
0x15DC	GPIO_TIMER3_ROUTEEN_SET	RW	TIMER3 pin enable
0x15E0	GPIO_TIMER3_CC0ROUTE_SET	RW	CC0 port/pin select
0x15E4	GPIO_TIMER3_CC1ROUTE_SET	RW	CC1 port/pin select
0x15E8	GPIO_TIMER3_CC2ROUTE_SET	RW	CC2 port/pin select
0x15EC	GPIO_TIMER3_CCC0ROUTE_SET	RW	CCC0 port/pin select
0x15F0	GPIO_TIMER3_CCC1ROUTE_SET	RW	CCC1 port/pin select
0x15F4	GPIO_TIMER3_CCC2ROUTE_SET	RW	CCC2 port/pin select
0x15FC	GPIO_TIMER4_ROUTEEN_SET	RW	TIMER4 pin enable
0x1600	GPIO_TIMER4_CC0ROUTE_SET	RW	CC0 port/pin select
0x1604	GPIO_TIMER4_CC1ROUTE_SET	RW	CC1 port/pin select
0x1608	GPIO_TIMER4_CC2ROUTE_SET	RW	CC2 port/pin select
0x160C	GPIO_TIMER4_CCC0ROUTE_SET	RW	CCC0 port/pin select
0x1610	GPIO_TIMER4_CCC1ROUTE_SET	RW	CCC1 port/pin select
0x1614	GPIO_TIMER4_CCC2ROUTE_SET	RW	CCC2 port/pin select
0x161C	GPIO_USART0_ROUTEEN_SET	RW	USART0 pin enable

Offset	Name	Type	Description
0x1620	GPIO_USART0_CSROUTE_SET	RW	CS port/pin select
0x1624	GPIO_USART0_CTSROUTE_SET	RW	CTS port/pin select
0x1628	GPIO_USART0_RTSROUTE_SET	RW	RTS port/pin select
0x162C	GPIO_USART0_RXROUTE_SET	RW	RX port/pin select
0x1630	GPIO_USART0_CLKROUTE_SET	RW	SCLK port/pin select
0x1634	GPIO_USART0_TXROUTE_SET	RW	TX port/pin select
0x163C	GPIO_USART1_ROUTEEN_SET	RW	USART1 pin enable
0x1640	GPIO_USART1_CSROUTE_SET	RW	CS port/pin select
0x1644	GPIO_USART1_CTSROUTE_SET	RW	CTS port/pin select
0x1648	GPIO_USART1_RTSROUTE_SET	RW	RTS port/pin select
0x164C	GPIO_USART1_RXROUTE_SET	RW	RX port/pin select
0x1650	GPIO_USART1_CLKROUTE_SET	RW	SCLK port/pin select
0x1654	GPIO_USART1_TXROUTE_SET	RW	TX port/pin select
0x2000	GPIO_PORTA_CTRL_CLR	RW	Port control
0x2004	GPIO_PORTA_MODEL_CLR	RW	mode low
0x200C	GPIO_PORTA_MODEH_CLR	RW	mode high
0x2010	GPIO_PORTA_DOUT_CLR	RW	data out
0x2014	GPIO_PORTA_DIN_CLR	RH	data in
0x2030	GPIO_PORTB_CTRL_CLR	RW	Port control
0x2034	GPIO_PORTB_MODEL_CLR	RW	mode low
0x2040	GPIO_PORTB_DOUT_CLR	RW	data out
0x2044	GPIO_PORTB_DIN_CLR	RH	data in
0x2060	GPIO_PORTC_CTRL_CLR	RW	Port control
0x2064	GPIO_PORTC_MODEL_CLR	RW	mode low
0x2070	GPIO_PORTC_DOUT_CLR	RW	data out
0x2074	GPIO_PORTC_DIN_CLR	RH	data in
0x2090	GPIO_PORTD_CTRL_CLR	RW	Port control
0x2094	GPIO_PORTD_MODEL_CLR	RW	mode low
0x20A0	GPIO_PORTD_DOUT_CLR	RW	data out
0x20A4	GPIO_PORTD_DIN_CLR	RH	data in
0x2300	GPIO_LOCK_CLR	W	Lock Register

Offset	Name	Type	Description
0x2310	GPIO_GPIOLOCKSTATUS_CLR	RH	Lock Status
0x2320	GPIO_ABUSALLOC_CLR	RW	A Bus allocation
0x2324	GPIO_BBUSALLOC_CLR	RW	B Bus allocation
0x2328	GPIO_CDBUSALLOC_CLR	RW	CD Bus allocation
0x2400	GPIO_EXTIPSELL_CLR	RW	External Interrupt Port Select Low
0x2404	GPIO_EXTIPSELH_CLR	RW	External interrupt Port Select High
0x2408	GPIO_EXTIPINSELL_CLR	RW	External Interrupt Pin Select Low
0x240C	GPIO_EXTIPINSELH_CLR	RW	External Interrupt Pin Select High
0x2410	GPIO_EXTIRISE_CLR	RW	External Interrupt Rising Edge Trigger
0x2414	GPIO_EXTIFALL_CLR	RW	External Interrupt Falling Edge Trigger
0x2420	GPIO_IF_CLR	RWH INTFLAG	Interrupt Flag
0x2424	GPIO_IEN_CLR	RW	Interrupt Enable
0x242C	GPIO_EM4WUEN_CLR	RW	EM4 wakeup enable
0x2430	GPIO_EM4WUPOL_CLR	RW	EM4 wakeup polarity
0x2440	GPIO_DBGROUTEPEN_CLR	RW	Debugger Route Pin enable
0x2444	GPIO_TRACEROUTEPEN_CLR	RW	Trace Route Pin Enable
0x2450	GPIO_CMU_ROUTEEN_CLR	RW	CMU pin enable
0x2454	GPIO_CMU_CLKIN0ROUTE_CLR	RW	CLKIN0 port/pin select
0x2458	GPIO_CMU_CLKOUT0ROUTE_CLR	RW	CLKOUT0 port/pin select
0x245C	GPIO_CMU_CLKOUT1ROUTE_CLR	RW	CLKOUT1 port/pin select
0x2460	GPIO_CMU_CLKOUT2ROUTE_CLR	RW	CLKOUT2 port/pin select
0x246C	GPIO_DCDC_ROUTEEN_CLR	RW	DCDC pin enable
0x247C	GPIO_FRC_ROUTEEN_CLR	RW	FRC pin enable
0x2480	GPIO_FRC_DCLKROUTE_CLR	RW	DCLK port/pin select
0x2484	GPIO_FRC_DFRAME-ROUTE_CLR	RW	DFRAME port/pin select
0x2488	GPIO_FRC_DOUTROUTE_CLR	RW	DOUT port/pin select
0x2490	GPIO_I2C0_ROUTEEN_CLR	RW	I2C0 pin enable
0x2494	GPIO_I2C0_SCLROUTE_CLR	RW	SCL port/pin select
0x2498	GPIO_I2C0_SDAROUTE_CLR	RW	SDA port/pin select
0x24A0	GPIO_I2C1_ROUTEEN_CLR	RW	I2C1 pin enable
0x24A4	GPIO_I2C1_SCLROUTE_CLR	RW	SCL port/pin select
0x24A8	GPIO_I2C1_SDAROUTE_CLR	RW	SDA port/pin select
0x24B0	GPIO_LETIMER0_ROU-TEEN_CLR	RW	LETIMER pin enable

Offset	Name	Type	Description
0x24B4	GPIO_LETIMER0_OUT0ROUTE_CLR	RW	OUT0 port/pin select
0x24B8	GPIO_LETIMER0_OUT1ROUTE_CLR	RW	OUT1 port/pin select
0x24C0	GPIO_EUART0_ROUTEEN_CLR	RW	EUART pin enable
0x24C4	GPIO_EUART0_CTSROUTE_CLR	RW	CTS port/pin select
0x24C8	GPIO_EUART0_RTSROUTE_CLR	RW	RTS port/pin select
0x24CC	GPIO_EUART0_RXROUTE_CLR	RW	RX port/pin select
0x24D0	GPIO_EUART0_TXROUTE_CLR	RW	TX port/pin select
0x24D8	GPIO_MODEM_ROUTEEN_CLR	RW	MODEM pin enable
0x24DC	GPIO_MODEM_ANT0ROUTE_CLR	RW	ANT0 port/pin select
0x24E0	GPIO_MODEM_ANT1ROUTE_CLR	RW	ANT1 port/pin select
0x24E4	GPIO_MODEM_ANTROLLOVERROUTE_CLR	RW	ANTROLLOVER port/pin select
0x24E8	GPIO_MODEM_ANTRR0ROUTE_CLR	RW	ANTRR0 port/pin select
0x24EC	GPIO_MODEM_ANTRR1ROUTE_CLR	RW	ANTRR1 port/pin select
0x24F0	GPIO_MODEM_ANTRR2ROUTE_CLR	RW	ANTRR2 port/pin select
0x24F4	GPIO_MODEM_ANTRR3ROUTE_CLR	RW	ANTRR3 port/pin select
0x24F8	GPIO_MODEM_ANTRR4ROUTE_CLR	RW	ANTRR4 port/pin select
0x24FC	GPIO_MODEM_ANTRR5ROUTE_CLR	RW	ANTRR5 port/pin select
0x2500	GPIO_MODEM_ANTSWENROUTE_CLR	RW	ANTSWEN port/pin select
0x2504	GPIO_MODEM_ANTSWUSROUTE_CLR	RW	ANTSWUS port/pin select
0x2508	GPIO_MODEM_ANTTRIGROUTE_CLR	RW	ANTTRIG port/pin select
0x250C	GPIO_MODEM_ANTTRIGSTOPROUTE_CLR	RW	ANTTRIGSTOP port/pin select
0x2510	GPIO_MODEM_DCLKROUTE_CLR	RW	DCLK port/pin select
0x2514	GPIO_MODEM_DINROUTE_CLR	RW	DIN port/pin select
0x2518	GPIO_MODEM_DOUTROUTE_CLR	RW	DOUT port/pin select

Offset	Name	Type	Description
0x2520	GPIO_PDM_ROUTEEN_CLR	RW	PDM pin enable
0x2524	GPIO_PDM_CLKROUTE_CLR	RW	CLK port/pin select
0x2528	GPIO_PDM_DAT0ROUTE_CLR	RW	DAT0 port/pin select
0x252C	GPIO_PDM_DAT1ROUTE_CLR	RW	DAT1 port/pin select
0x2534	GPIO_PRS0_ROUTEEN_CLR	RW	PRS0 pin enable
0x2538	GPIO_PRS0_ASYNCH0ROUTE_CLR	RW	ASYNCH0 port/pin select
0x253C	GPIO_PRS0_ASYNCH1ROUTE_CLR	RW	ASYNCH1 port/pin select
0x2540	GPIO_PRS0_ASYNCH2ROUTE_CLR	RW	ASYNCH2 port/pin select
0x2544	GPIO_PRS0_ASYNCH3ROUTE_CLR	RW	ASYNCH3 port/pin select
0x2548	GPIO_PRS0_ASYNCH4ROUTE_CLR	RW	ASYNCH4 port/pin select
0x254C	GPIO_PRS0_ASYNCH5ROUTE_CLR	RW	ASYNCH5 port/pin select
0x2550	GPIO_PRS0_ASYNCH6ROUTE_CLR	RW	ASYNCH6 port/pin select
0x2554	GPIO_PRS0_ASYNCH7ROUTE_CLR	RW	ASYNCH7 port/pin select
0x2558	GPIO_PRS0_ASYNCH8ROUTE_CLR	RW	ASYNCH8 port/pin select
0x255C	GPIO_PRS0_ASYNCH9ROUTE_CLR	RW	ASYNCH9 port/pin select
0x2560	GPIO_PRS0_ASYNCH10ROUTE_CLR	RW	ASYNCH10 port/pin select
0x2564	GPIO_PRS0_ASYNCH11ROUTE_CLR	RW	ASYNCH11 port/pin select
0x2568	GPIO_PRS0_SYNCH0ROUTE_CLR	RW	SYNCH0 port/pin select
0x256C	GPIO_PRS0_SYNCH1ROUTE_CLR	RW	SYNCH1 port/pin select
0x2570	GPIO_PRS0_SYNCH2ROUTE_CLR	RW	SYNCH2 port/pin select
0x2574	GPIO_PRS0_SYNCH3ROUTE_CLR	RW	SYNCH3 port/pin select
0x257C	GPIO_TIMER0_ROUTEEN_CLR	RW	TIMER0 pin enable
0x2580	GPIO_TIMER0_CC0ROUTE_CLR	RW	CC0 port/pin select
0x2584	GPIO_TIMER0_CC1ROUTE_CLR	RW	CC1 port/pin select
0x2588	GPIO_TIMER0_CC2ROUTE_CLR	RW	CC2 port/pin select

Offset	Name	Type	Description
0x258C	GPIO_TIMER0_CCC0ROUTE_CLR	RW	CCC0 port/pin select
0x2590	GPIO_TIMER0_CCC1ROUTE_CLR	RW	CCC1 port/pin select
0x2594	GPIO_TIMER0_CCC2ROUTE_CLR	RW	CCC2 port/pin select
0x259C	GPIO_TIMER1_ROUTEEN_CLR	RW	TIMER1 pin enable
0x25A0	GPIO_TIMER1_CC0ROUTE_CLR	RW	CC0 port/pin select
0x25A4	GPIO_TIMER1_CC1ROUTE_CLR	RW	CC1 port/pin select
0x25A8	GPIO_TIMER1_CC2ROUTE_CLR	RW	CC2 port/pin select
0x25AC	GPIO_TIMER1_CCC0ROUTE_CLR	RW	CCC0 port/pin select
0x25B0	GPIO_TIMER1_CCC1ROUTE_CLR	RW	CCC1 port/pin select
0x25B4	GPIO_TIMER1_CCC2ROUTE_CLR	RW	CCC2 port/pin select
0x25BC	GPIO_TIMER2_ROUTEEN_CLR	RW	TIMER2 pin enable
0x25C0	GPIO_TIMER2_CC0ROUTE_CLR	RW	CC0 port/pin select
0x25C4	GPIO_TIMER2_CC1ROUTE_CLR	RW	CC1 port/pin select
0x25C8	GPIO_TIMER2_CC2ROUTE_CLR	RW	CC2 port/pin select
0x25CC	GPIO_TIMER2_CCC0ROUTE_CLR	RW	CCC0 port/pin select
0x25D0	GPIO_TIMER2_CCC1ROUTE_CLR	RW	CCC1 port/pin select
0x25D4	GPIO_TIMER2_CCC2ROUTE_CLR	RW	CCC2 port/pin select
0x25DC	GPIO_TIMER3_ROUTEEN_CLR	RW	TIMER3 pin enable
0x25E0	GPIO_TIMER3_CC0ROUTE_CLR	RW	CC0 port/pin select
0x25E4	GPIO_TIMER3_CC1ROUTE_CLR	RW	CC1 port/pin select
0x25E8	GPIO_TIMER3_CC2ROUTE_CLR	RW	CC2 port/pin select
0x25EC	GPIO_TIMER3_CCC0ROUTE_CLR	RW	CCC0 port/pin select
0x25F0	GPIO_TIMER3_CCC1ROUTE_CLR	RW	CCC1 port/pin select
0x25F4	GPIO_TIMER3_CCC2ROUTE_CLR	RW	CCC2 port/pin select
0x25FC	GPIO_TIMER4_ROUTEEN_CLR	RW	TIMER4 pin enable

Offset	Name	Type	Description
0x2600	GPIO_TIMER4_CC0ROUTE_CLR	RW	CC0 port/pin select
0x2604	GPIO_TIMER4_CC1ROUTE_CLR	RW	CC1 port/pin select
0x2608	GPIO_TIMER4_CC2ROUTE_CLR	RW	CC2 port/pin select
0x260C	GPIO_TIMER4_CCC0ROUTE_CLR	RW	CCC0 port/pin select
0x2610	GPIO_TIMER4_CCC1ROUTE_CLR	RW	CCC1 port/pin select
0x2614	GPIO_TIMER4_CCC2ROUTE_CLR	RW	CCC2 port/pin select
0x261C	GPIO_USART0_ROUTEEN_CLR	RW	USART0 pin enable
0x2620	GPIO_USART0_CSROUTE_CLR	RW	CS port/pin select
0x2624	GPIO_USART0_CTSROUTE_CLR	RW	CTS port/pin select
0x2628	GPIO_USART0_RTSROUTE_CLR	RW	RTS port/pin select
0x262C	GPIO_USART0_RXROUTE_CLR	RW	RX port/pin select
0x2630	GPIO_USART0_CLKROUTE_CLR	RW	SCLK port/pin select
0x2634	GPIO_USART0_TXROUTE_CLR	RW	TX port/pin select
0x263C	GPIO_USART1_ROUTEEN_CLR	RW	USART1 pin enable
0x2640	GPIO_USART1_CSROUTE_CLR	RW	CS port/pin select
0x2644	GPIO_USART1_CTSROUTE_CLR	RW	CTS port/pin select
0x2648	GPIO_USART1_RTSROUTE_CLR	RW	RTS port/pin select
0x264C	GPIO_USART1_RXROUTE_CLR	RW	RX port/pin select
0x2650	GPIO_USART1_CLKROUTE_CLR	RW	SCLK port/pin select
0x2654	GPIO_USART1_TXROUTE_CLR	RW	TX port/pin select
0x3000	GPIO_PORTA_CTRL_TGL	RW	Port control
0x3004	GPIO_PORTA_MODEL_TGL	RW	mode low
0x300C	GPIO_PORTA_MODEH_TGL	RW	mode high
0x3010	GPIO_PORTA_DOUT_TGL	RW	data out
0x3014	GPIO_PORTA_DIN_TGL	RH	data in
0x3030	GPIO_PORTB_CTRL_TGL	RW	Port control

Offset	Name	Type	Description
0x3034	GPIO_PORTB_MODEL_TGL	RW	mode low
0x3040	GPIO_PORTB_DOUT_TGL	RW	data out
0x3044	GPIO_PORTB_DIN_TGL	RH	data in
0x3060	GPIO_PORTC_CTRL_TGL	RW	Port control
0x3064	GPIO_PORTC_MODEL_TGL	RW	mode low
0x3070	GPIO_PORTC_DOUT_TGL	RW	data out
0x3074	GPIO_PORTC_DIN_TGL	RH	data in
0x3090	GPIO_PORTD_CTRL_TGL	RW	Port control
0x3094	GPIO_PORTD_MODEL_TGL	RW	mode low
0x30A0	GPIO_PORTD_DOUT_TGL	RW	data out
0x30A4	GPIO_PORTD_DIN_TGL	RH	data in
0x3300	GPIO_LOCK_TGL	W	Lock Register
0x3310	GPIO_GPIOLOCKSTATUS_TGL	RH	Lock Status
0x3320	GPIO_ABUSALLOC_TGL	RW	A Bus allocation
0x3324	GPIO_BBUSALLOC_TGL	RW	B Bus allocation
0x3328	GPIO_CDBUSALLOC_TGL	RW	CD Bus allocation
0x3400	GPIO_EXTIPSELL_TGL	RW	External Interrupt Port Select Low
0x3404	GPIO_EXTIPSELH_TGL	RW	External interrupt Port Select High
0x3408	GPIO_EXTIPINSELL_TGL	RW	External Interrupt Pin Select Low
0x340C	GPIO_EXTIPINSELH_TGL	RW	External Interrupt Pin Select High
0x3410	GPIO_EXTIRISE_TGL	RW	External Interrupt Rising Edge Trigger
0x3414	GPIO_EXTIFALL_TGL	RW	External Interrupt Falling Edge Trigger
0x3420	GPIO_IF_TGL	RWH INTFLAG	Interrupt Flag
0x3424	GPIO_IEN_TGL	RW	Interrupt Enable
0x342C	GPIO_EM4WUEN_TGL	RW	EM4 wakeup enable
0x3430	GPIO_EM4WUPOL_TGL	RW	EM4 wakeup polarity
0x3440	GPIO_DBGROUTEPEN_TGL	RW	Debugger Route Pin enable
0x3444	GPIO_TRACEROUTEPEN_TGL	RW	Trace Route Pin Enable
0x3450	GPIO_CMU_ROUTEEN_TGL	RW	CMU pin enable
0x3454	GPIO_CMU_CLKIN0ROUTE_TGL	RW	CLKIN0 port/pin select
0x3458	GPIO_CMU_CLKOUT0ROUTE_TGL	RW	CLKOUT0 port/pin select
0x345C	GPIO_CMU_CLKOUT1ROUTE_TGL	RW	CLKOUT1 port/pin select
0x3460	GPIO_CMU_CLKOUT2ROUTE_TGL	RW	CLKOUT2 port/pin select
0x346C	GPIO_DCDC_ROUTEEN_TGL	RW	DCDC pin enable
0x347C	GPIO_FRC_ROUTEEN_TGL	RW	FRC pin enable

Offset	Name	Type	Description
0x3480	GPIO_FRC_DCLKROUTE_TGL	RW	DCLK port/pin select
0x3484	GPIO_FRC_DFRAME-ROUTE_TGL	RW	DFRAME port/pin select
0x3488	GPIO_FRC_DOUTROUTE_TGL	RW	DOUT port/pin select
0x3490	GPIO_I2C0_ROUTEEN_TGL	RW	I2C0 pin enable
0x3494	GPIO_I2C0_SCLROUTE_TGL	RW	SCL port/pin select
0x3498	GPIO_I2C0_SDAROUTE_TGL	RW	SDA port/pin select
0x34A0	GPIO_I2C1_ROUTEEN_TGL	RW	I2C1 pin enable
0x34A4	GPIO_I2C1_SCLROUTE_TGL	RW	SCL port/pin select
0x34A8	GPIO_I2C1_SDAROUTE_TGL	RW	SDA port/pin select
0x34B0	GPIO_LETIMER0_ROU-TEEN_TGL	RW	LETIMER pin enable
0x34B4	GPIO_LETIM-ER0_OUT0ROUTE_TGL	RW	OUT0 port/pin select
0x34B8	GPIO_LETIM-ER0_OUT1ROUTE_TGL	RW	OUT1 port/pin select
0x34C0	GPIO_EUART0_ROU-TEEN_TGL	RW	EUART pin enable
0x34C4	GPIO_EU-ART0_CTSROUTE_TGL	RW	CTS port/pin select
0x34C8	GPIO_EU-ART0_RTSROUTE_TGL	RW	RTS port/pin select
0x34CC	GPIO_EU-ART0_RXROUTE_TGL	RW	RX port/pin select
0x34D0	GPIO_EUART0_TXROUTE_TGL	RW	TX port/pin select
0x34D8	GPIO_MODEM_ROUTEEN_TGL	RW	MODEM pin enable
0x34DC	GPIO_MO-DEM_ANT0ROUTE_TGL	RW	ANT0 port/pin select
0x34E0	GPIO_MO-DEM_ANT1ROUTE_TGL	RW	ANT1 port/pin select
0x34E4	GPIO_MODEM_ANTROLLO-VERROUTE_TGL	RW	ANTROLLOVER port/pin select
0x34E8	GPIO_MO-DEM_ANTRR0ROUTE_TGL	RW	ANTRR0 port/pin select
0x34EC	GPIO_MO-DEM_ANTRR1ROUTE_TGL	RW	ANTRR1 port/pin select
0x34F0	GPIO_MO-DEM_ANTRR2ROUTE_TGL	RW	ANTRR2 port/pin select
0x34F4	GPIO_MO-DEM_ANTRR3ROUTE_TGL	RW	ANTRR3 port/pin select
0x34F8	GPIO_MO-DEM_ANTRR4ROUTE_TGL	RW	ANTRR4 port/pin select
0x34FC	GPIO_MO-DEM_ANTRR5ROUTE_TGL	RW	ANTRR5 port/pin select

Offset	Name	Type	Description
0x3500	GPIO_MODEM_ANTSWEN-ROUTE_TGL	RW	ANTSWEN port/pin select
0x3504	GPIO_MODEM_ANTSWUS-ROUTE_TGL	RW	ANTSWUS port/pin select
0x3508	GPIO_MODEM_ANTTRIG-ROUTE_TGL	RW	ANTTRIG port/pin select
0x350C	GPIO_MODEM_ANTTRIGSTOP-ROUTE_TGL	RW	ANTTRIGSTOP port/pin select
0x3510	GPIO_MO-DEM_DCLKROUTE_TGL	RW	DCLK port/pin select
0x3514	GPIO_MODEM_DIN-ROUTE_TGL	RW	DIN port/pin select
0x3518	GPIO_MODEM_DOUT-ROUTE_TGL	RW	DOUT port/pin select
0x3520	GPIO_PDM_ROUTEEN_TGL	RW	PDM pin enable
0x3524	GPIO_PDM_CLKROUTE_TGL	RW	CLK port/pin select
0x3528	GPIO_PDM_DAT0ROUTE_TGL	RW	DAT0 port/pin select
0x352C	GPIO_PDM_DAT1ROUTE_TGL	RW	DAT1 port/pin select
0x3534	GPIO_PRS0_ROUTEEN_TGL	RW	PRS0 pin enable
0x3538	GPIO_PRS0_ASYNCH0ROUTE_TGL	RW	ASYNCH0 port/pin select
0x353C	GPIO_PRS0_ASYNCH1ROUTE_TGL	RW	ASYNCH1 port/pin select
0x3540	GPIO_PRS0_ASYNCH2ROUTE_TGL	RW	ASYNCH2 port/pin select
0x3544	GPIO_PRS0_ASYNCH3ROUTE_TGL	RW	ASYNCH3 port/pin select
0x3548	GPIO_PRS0_ASYNCH4ROUTE_TGL	RW	ASYNCH4 port/pin select
0x354C	GPIO_PRS0_ASYNCH5ROUTE_TGL	RW	ASYNCH5 port/pin select
0x3550	GPIO_PRS0_ASYNCH6ROUTE_TGL	RW	ASYNCH6 port/pin select
0x3554	GPIO_PRS0_ASYNCH7ROUTE_TGL	RW	ASYNCH7 port/pin select
0x3558	GPIO_PRS0_ASYNCH8ROUTE_TGL	RW	ASYNCH8 port/pin select
0x355C	GPIO_PRS0_ASYNCH9ROUTE_TGL	RW	ASYNCH9 port/pin select
0x3560	GPIO_PRS0_ASYNCH10ROUTE_TGL	RW	ASYNCH10 port/pin select
0x3564	GPIO_PRS0_ASYNCH11ROUTE_TGL	RW	ASYNCH11 port/pin select
0x3568	GPIO_PRS0_SYNCH0ROUTE_TGL	RW	SYNCH0 port/pin select

Offset	Name	Type	Description
0x356C	GPIO_PRS0_SYNCH1ROUTE_TGL	RW	SYNCH1 port/pin select
0x3570	GPIO_PRS0_SYNCH2ROUTE_TGL	RW	SYNCH2 port/pin select
0x3574	GPIO_PRS0_SYNCH3ROUTE_TGL	RW	SYNCH3 port/pin select
0x357C	GPIO_TIMER0_ROUTEEN_TGL	RW	TIMER0 pin enable
0x3580	GPIO_TIMER0_CC0ROUTE_TGL	RW	CC0 port/pin select
0x3584	GPIO_TIMER0_CC1ROUTE_TGL	RW	CC1 port/pin select
0x3588	GPIO_TIMER0_CC2ROUTE_TGL	RW	CC2 port/pin select
0x358C	GPIO_TIMER0_CCC0ROUTE_TGL	RW	CCC0 port/pin select
0x3590	GPIO_TIMER0_CCC1ROUTE_TGL	RW	CCC1 port/pin select
0x3594	GPIO_TIMER0_CCC2ROUTE_TGL	RW	CCC2 port/pin select
0x359C	GPIO_TIMER1_ROUTEEN_TGL	RW	TIMER1 pin enable
0x35A0	GPIO_TIMER1_CC0ROUTE_TGL	RW	CC0 port/pin select
0x35A4	GPIO_TIMER1_CC1ROUTE_TGL	RW	CC1 port/pin select
0x35A8	GPIO_TIMER1_CC2ROUTE_TGL	RW	CC2 port/pin select
0x35AC	GPIO_TIMER1_CCC0ROUTE_TGL	RW	CCC0 port/pin select
0x35B0	GPIO_TIMER1_CCC1ROUTE_TGL	RW	CCC1 port/pin select
0x35B4	GPIO_TIMER1_CCC2ROUTE_TGL	RW	CCC2 port/pin select
0x35BC	GPIO_TIMER2_ROUTEEN_TGL	RW	TIMER2 pin enable
0x35C0	GPIO_TIMER2_CC0ROUTE_TGL	RW	CC0 port/pin select
0x35C4	GPIO_TIMER2_CC1ROUTE_TGL	RW	CC1 port/pin select
0x35C8	GPIO_TIMER2_CC2ROUTE_TGL	RW	CC2 port/pin select
0x35CC	GPIO_TIMER2_CCC0ROUTE_TGL	RW	CCC0 port/pin select
0x35D0	GPIO_TIMER2_CCC1ROUTE_TGL	RW	CCC1 port/pin select
0x35D4	GPIO_TIMER2_CCC2ROUTE_TGL	RW	CCC2 port/pin select
0x35DC	GPIO_TIMER3_ROUTEEN_TGL	RW	TIMER3 pin enable

Offset	Name	Type	Description
0x35E0	GPIO_TIMER3_CC0ROUTE_TGL	RW	CC0 port/pin select
0x35E4	GPIO_TIMER3_CC1ROUTE_TGL	RW	CC1 port/pin select
0x35E8	GPIO_TIMER3_CC2ROUTE_TGL	RW	CC2 port/pin select
0x35EC	GPIO_TIMER3_CCC0ROUTE_TGL	RW	CCC0 port/pin select
0x35F0	GPIO_TIMER3_CCC1ROUTE_TGL	RW	CCC1 port/pin select
0x35F4	GPIO_TIMER3_CCC2ROUTE_TGL	RW	CCC2 port/pin select
0x35FC	GPIO_TIMER4_ROUTEEN_TGL	RW	TIMER4 pin enable
0x3600	GPIO_TIMER4_CC0ROUTE_TGL	RW	CC0 port/pin select
0x3604	GPIO_TIMER4_CC1ROUTE_TGL	RW	CC1 port/pin select
0x3608	GPIO_TIMER4_CC2ROUTE_TGL	RW	CC2 port/pin select
0x360C	GPIO_TIMER4_CCC0ROUTE_TGL	RW	CCC0 port/pin select
0x3610	GPIO_TIMER4_CCC1ROUTE_TGL	RW	CCC1 port/pin select
0x3614	GPIO_TIMER4_CCC2ROUTE_TGL	RW	CCC2 port/pin select
0x361C	GPIO_USART0_ROUTEEN_TGL	RW	USART0 pin enable
0x3620	GPIO_USART0_CSROUTE_TGL	RW	CS port/pin select
0x3624	GPIO_USART0_CTSROUTE_TGL	RW	CTS port/pin select
0x3628	GPIO_USART0_RTSROUTE_TGL	RW	RTS port/pin select
0x362C	GPIO_USART0_RXROUTE_TGL	RW	RX port/pin select
0x3630	GPIO_USART0_CLKROUTE_TGL	RW	SCLK port/pin select
0x3634	GPIO_USART0_TXROUTE_TGL	RW	TX port/pin select
0x363C	GPIO_USART1_ROUTEEN_TGL	RW	USART1 pin enable
0x3640	GPIO_USART1_CSROUTE_TGL	RW	CS port/pin select
0x3644	GPIO_USART1_CTSROUTE_TGL	RW	CTS port/pin select
0x3648	GPIO_USART1_RTSROUTE_TGL	RW	RTS port/pin select

Offset	Name	Type	Description
0x364C	GPIO_USART1_RXROUTE_TGL	RW	RX port/pin select
0x3650	GPIO_USART1_CLKROUTE_TGL	RW	SCLK port/pin select
0x3654	GPIO_USART1_TXROUTE_TGL	RW	TX port/pin select

25.6 GPIO Register Description

25.6.1 GPIO_PORTA_CTRL - Port control

Offset	Bit Position																														
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0							0x4									0x0							0x4					
Access				RW							RW									RW							RW					
Name				DINDISALT							SLEWRATEALT									DINDIS							SLEWRATE					

Bit	Name	Reset	Access	Description
31:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28	DINDISALT	0x0	RW	**Data In Disable Alt**
	Data input disable for port pins using alternate modes.			
27:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	SLEWRATEALT	0x4	RW	**Slew Rate Alt**
	Slewrate limit for port pins using alternate modes. Higher values provide faster slew rates.			
19:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	DINDIS	0x0	RW	**Data In Disable**
	Data input disable for port pins not using alternate modes.			
11:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	SLEWRATE	0x4	RW	**Slew Rate**
	Slewrate limit for port pins not using alternate modes. Higher values provide faster slew rates.			
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

25.6.2 GPIO_PORTA_MODEL - mode low

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0				0x0				0x0				0x0				0x0				0x0				0x0				0x0			
Access	RW				RW				RW				RW				RW				RW				RW				RW			
Name	MODE7				MODE6				MODE5				MODE4				MODE3				MODE2				MODE1				MODE0			

Bit	Name	Reset	Access	Description
31:28	MODE7	0x0	RW	**MODE n**
	MODE n			

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFIL-TER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUP-FILTER	Open-drain output using alternate control with filter and pullup.

27:24	MODE6	0x0	RW	**MODE n**
	MODE n			

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.

Bit	Name	Reset	Access	Description
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
23:20	MODE5 MODE n	0x0	RW	**MODE n**
	Value	Mode		Description
	0	DISABLED		Input disabled. Pullup if DOUT is set.
	1	INPUT		Input enabled. Filter if DOUT is set.
	2	INPUTPULL		Input enabled. DOUT determines pull direction.
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
19:16	MODE4 MODE n	0x0	RW	**MODE n**

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	DISABLED		Input disabled. Pullup if DOUT is set.
	1	INPUT		Input enabled. Filter if DOUT is set.
	2	INPUTPULL		Input enabled. DOUT determines pull direction.
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
15:12	MODE3 MODE n	0x0	RW	**MODE n**
	Value	Mode		Description
	0	DISABLED		Input disabled. Pullup if DOUT is set.
	1	INPUT		Input enabled. Filter if DOUT is set.
	2	INPUTPULL		Input enabled. DOUT determines pull direction.
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.

Bit	Name	Reset	Access	Description
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
11:8	MODE2 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFIL-TER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUP-FILTER	Open-drain output using alternate control with filter and pullup.

7:4	MODE1 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.

Bit	Name	Reset	Access	Description
	11	WIREDANDPULLUPFILTER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUPFILTER		Open-drain output using alternate control with filter and pullup.
3:0	MODE0 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFILTER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUPFILTER	Open-drain output using alternate control with filter and pullup.

25.6.3 GPIO_PORTA_MODEH - mode high

Offset	Bit Position																															
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													MODE0			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	MODE0 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFILTER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUPFILTER	Open-drain output using alternate control with filter and pullup.

25.6.4 GPIO_PORTA_DOUT - data out

Offset	Bit Position																																
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												RW				
Name																												DOUT				

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	DOUT	0x0	RW	**Data output**
	Data output			

25.6.5 GPIO_PORTA_DIN - data in

Offset	Bit Position																																
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												R				
Name																												DIN				

Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	DIN	0x0	R	**Data input**
	Data input			

25.6.6 GPIO_PORTB_CTRL - Port control

Offset	Bit Position																															
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0							0x4									0x0							0x4						
Access				RW							RW									RW							RW						
Name				DINDISALT							SLEWRATEALT									DINDIS							SLEWRATE						

Bit	Name	Reset	Access	Description
31:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28	DINDISALT	0x0	RW	**Data In Disable Alt**
	Data input disable for port pins using alternate modes.			
27:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	SLEWRATEALT	0x4	RW	**Slew Rate Alt**
	Slewrate limit for port pins using alternate modes. Higher values representer faster slewrates.			
19:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	DINDIS	0x0	RW	**Data In Disable**
	Data input disable for port pins not using alternate modes.			
11:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	SLEWRATE	0x4	RW	**Slew Rate**
	Slewrate limit for port pins using not alternate modes. Higher values representer faster slewrates.			
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

25.6.7 GPIO_PORTB_MODEL - mode low

Offset	Bit Position																													
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0				0x0				0x0				0x0				0x0			
Access													RW				RW				RW				RW				RW			
Name													MODE4				MODE3				MODE2				MODE1				MODE0			

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	MODE4 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFILTER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUPFILTER	Open-drain output using alternate control with filter and pullup.

15:12	MODE3 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.

Bit	Name	Reset	Access	Description
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
11:8	MODE2 MODE n	0x0	RW	**MODE n**
	Value	Mode		Description
	0	DISABLED		Input disabled. Pullup if DOUT is set.
	1	INPUT		Input enabled. Filter if DOUT is set.
	2	INPUTPULL		Input enabled. DOUT determines pull direction.
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
7:4	MODE1 MODE n	0x0	RW	**MODE n**

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFIL-TER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUP-FILTER	Open-drain output using alternate control with filter and pullup.

3:0	MODE0 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFIL-TER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.

Bit	Name	Reset	Access	Description
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.

25.6.8 GPIO_PORTB_DOUT - data out

Offset	Bit Position																															
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0		
Access																														RW		
Name																														DOUT		

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4:0	DOUT Data output	0x0	RW	**Data output**

25.6.9 GPIO_PORTB_DIN - data in

Offset	Bit Position																															
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0		
Access																														R		
Name																														DIN		

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4:0	DIN Data input	0x0	R	**Data input**

25.6.10 GPIO_PORTC_CTRL - Port control

Offset	Bit Position																													
0x060	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0							0x4									0x0							0x4					
Access				RW							RW									RW							RW					
Name				DINDISALT							SLEWRATEALT									DINDIS							SLEWRATE					

Bit	Name	Reset	Access	Description
31:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28	DINDISALT	0x0	RW	**Data In Disable Alt**
	Data input disable for port pins using alternate modes.			
27:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	SLEWRATEALT	0x4	RW	**Slew Rate Alt**
	Slewrate limit for port pins using alternate modes. Higher values representer faster slewrates.			
19:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	DINDIS	0x0	RW	**Data In Disable**
	Data input disable for port pins not using alternate modes.			
11:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	SLEWRATE	0x4	RW	**Slew Rate**
	Slewrate limit for port pins using not alternate modes. Higher values representer faster slewrates.			
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

25.6.11 GPIO_PORTC_MODEL - mode low

Offset	Bit Position																													
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0				0x0				0x0				0x0				0x0				0x0				0x0				0x0			
Access	RW				RW				RW				RW				RW				RW				RW				RW			
Name	MODE7				MODE6				MODE5				MODE4				MODE3				MODE2				MODE1				MODE0			

Bit	Name	Reset	Access	Description
31:28	MODE7	0x0	RW	**MODE n**
	MODE n			

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFIL-TER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUP-FILTER	Open-drain output using alternate control with filter and pullup.

27:24	MODE6	0x0	RW	**MODE n**
	MODE n			

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.

Bit	Name	Reset	Access	Description
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
23:20	MODE5 MODE n	0x0	RW	**MODE n**
	Value	Mode		Description
	0	DISABLED		Input disabled. Pullup if DOUT is set.
	1	INPUT		Input enabled. Filter if DOUT is set.
	2	INPUTPULL		Input enabled. DOUT determines pull direction.
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
19:16	MODE4 MODE n	0x0	RW	**MODE n**

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	DISABLED		Input disabled. Pullup if DOUT is set.
	1	INPUT		Input enabled. Filter if DOUT is set.
	2	INPUTPULL		Input enabled. DOUT determines pull direction.
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
15:12	MODE3 MODE n	0x0	RW	**MODE n**
	Value	Mode		Description
	0	DISABLED		Input disabled. Pullup if DOUT is set.
	1	INPUT		Input enabled. Filter if DOUT is set.
	2	INPUTPULL		Input enabled. DOUT determines pull direction.
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.

Bit	Name	Reset	Access	Description
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
11:8	MODE2 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFIL-TER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUP-FILTER	Open-drain output using alternate control with filter and pullup.

7:4	MODE1 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.

Bit	Name	Reset	Access	Description
	11	WIREDANDPULLUPFILTER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUPFILTER		Open-drain output using alternate control with filter and pullup.
3:0	MODE0 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFILTER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUPFILTER	Open-drain output using alternate control with filter and pullup.

25.6.12 GPIO_PORTC_DOUT - data out

Offset	Bit Position																													
0x070	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									RW							
Name																									DOUT							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	DOUT	0x0	RW	**Data output**
	Data output			

25.6.13 GPIO_PORTC_DIN - data in

Offset	Bit Position																													
0x074	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									DIN							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	DIN	0x0	R	**Data input**
	Data input			

25.6.14 GPIO_PORTD_CTRL - Port control

Offset	Bit Position																													
0x090	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0							0x4									0x0							0x4					
Access				RW							RW									RW							RW					
Name				DINDISALT							SLEWRATEALT									DINDIS							SLEWRATE					

Bit	Name	Reset	Access	Description
31:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28	DINDISALT	0x0	RW	**Data In Disable Alt**
	Data input disable for port pins using alternate modes.			
27:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	SLEWRATEALT	0x4	RW	**Slew Rate Alt**
	Slewrate limit for port pins using alternate modes. Higher values representer faster slewrates.			
19:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	DINDIS	0x0	RW	**Data In Disable**
	Data input disable for port pins not using alternate modes.			
11:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	SLEWRATE	0x4	RW	**Slew Rate**
	Slewrate limit for port pins using not alternate modes. Higher values representer faster slewrates.			
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

25.6.15 GPIO_PORTD_MODEL - mode low

Offset	Bit Position																													
0x094	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0				0x0				0x0				0x0			
Access																	RW				RW				RW				RW			
Name																	MODE3				MODE2				MODE1				MODE0			

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:12	MODE3 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFILTER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUPFILTER	Open-drain output using alternate control with filter and pullup.

11:8	MODE2 MODE n	0x0	RW	**MODE n**

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.

Bit	Name	Reset	Access	Description
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
7:4	MODE1 MODE n	0x0	RW	**MODE n**
	Value	Mode		Description
	0	DISABLED		Input disabled. Pullup if DOUT is set.
	1	INPUT		Input enabled. Filter if DOUT is set.
	2	INPUTPULL		Input enabled. DOUT determines pull direction.
	3	INPUTPULLFILTER		Input enabled with filter. DOUT determines pull direction.
	4	PUSHPULL		Push-pull output.
	5	PUSHPULLALT		Push-pull using alternate control.
	6	WIREDOR		Wired-or output.
	7	WIREDORPULLDOWN		Wired-or output with pull-down.
	8	WIREDAND		Open-drain output.
	9	WIREDANDFILTER		Open-drain output with filter.
	10	WIREDANDPULLUP		Open-drain output with pullup.
	11	WIREDANDPULLUPFIL-TER		Open-drain output with filter and pullup.
	12	WIREDANDALT		Open-drain output using alternate control.
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter.
	14	WIREDANDALTPULLUP		Open-drain output using alternate control with pullup.
	15	WIREDANDALTPULLUP-FILTER		Open-drain output using alternate control with filter and pullup.
3:0	MODE0 MODE n	0x0	RW	**MODE n**

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	DISABLED	Input disabled. Pullup if DOUT is set.
1	INPUT	Input enabled. Filter if DOUT is set.
2	INPUTPULL	Input enabled. DOUT determines pull direction.
3	INPUTPULLFILTER	Input enabled with filter. DOUT determines pull direction.
4	PUSHPULL	Push-pull output.
5	PUSHPULLALT	Push-pull using alternate control.
6	WIREDOR	Wired-or output.
7	WIREDORPULLDOWN	Wired-or output with pull-down.
8	WIREDAND	Open-drain output.
9	WIREDANDFILTER	Open-drain output with filter.
10	WIREDANDPULLUP	Open-drain output with pullup.
11	WIREDANDPULLUPFIL-TER	Open-drain output with filter and pullup.
12	WIREDANDALT	Open-drain output using alternate control.
13	WIREDANDALTFILTER	Open-drain output using alternate control with filter.
14	WIREDANDALTPULLUP	Open-drain output using alternate control with pullup.
15	WIREDANDALTPULLUP-FILTER	Open-drain output using alternate control with filter and pullup.

25.6.16 GPIO_PORTD_DOUT - data out

Offset	Bit Position																															
0x0A0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0			
Access																													RW			
Name																													DOUT			

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	DOUT Data output	0x0	RW	**Data output**

25.6.17 GPIO_PORTD_DIN - data in

Offset	Bit Position																													
0x0A4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x0		
Access																													R		
Name																													DIN		

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	DIN Data input	0x0	R	**Data input**

25.6.18 GPIO_LOCK - Lock Register

Offset	Bit Position																													
0x300	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0xA534															
Access																	W															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0xA534	W	**Configuration Lock Key**

Write any other value than the unlock code to lock configuration registers. Write the unlock code to unlock (See text for detailed list of configuration registers.)

Value	Mode	Description
42292	UNLOCK	Unlock code

25.6.19 GPIO_GPIOLOCKSTATUS - Lock Status

Offset	Bit Position																													
0x310	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																R
Name																																LOCK

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	LOCK	0x0	R	**GPIO LOCK status**
	Indicates current lock status of GPIO registers			

Value	Mode	Description
0	UNLOCKED	Registers are unlocked
1	LOCKED	Registers are locked

25.6.20 GPIO_ABUSALLOC - A Bus allocation

Offset	Bit Position																													
0x320	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0								0x0								0x0								0x0			
Access					RW								RW								RW								RW			
Name					AODD1								AODD0								AEVEN1								AEVEN0			

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:24	AODD1	0x0	RW	**A Bus Odd 1**
	peripheral allocation to A Bus Odd 1			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	AODD0	0x0	RW	**A Bus Odd 0**
	peripheral allocation to A Bus Odd 0			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	AEVEN1	0x0	RW	**A Bus Even 1**
	peripheral allocation to A Bus Even 1			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	AEVEN0	0x0	RW	**A Bus Even 0**
	peripheral allocation to A Bus Even 0			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0

Bit	Name	Reset	Access	Description

25.6.21 GPIO_BBUSALLOC - B Bus allocation

Offset	Bit Position																													
0x324	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0								0x0								0x0								0x0			
Access					RW								RW								RW								RW			
Name					BODD1								BODD0								BEVEN1								BEVEN0			

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:24	BODD1	0x0	RW	**B Bus Odd 1**
	peripheral allocation to B Bus Odd 1			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	BODD0	0x0	RW	**B Bus Odd 0**
	peripheral allocation to B Bus Odd 0			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	BEVEN1	0x0	RW	**B Bus Even 1**
	peripheral allocation to B Bus Even 1			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	BEVEN0	0x0	RW	**B Bus Even 0**
	peripheral allocation to B Bus Even 0			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0

Bit	Name	Reset	Access	Description

25.6.22 GPIO_CDBUSALLOC - CD Bus allocation

Offset	Bit Position																													
0x328	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0								0x0								0x0								0x0			
Access					RW								RW								RW								RW			
Name					CDODD1								CDODD0								CDEVEN1								CDEVEN0			

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:24	CDODD1	0x0	RW	**CD Bus Odd 1**
	peripheral allocation to CD Bus Odd 1			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	CDODD0	0x0	RW	**CD Bus Odd 0**
	peripheral allocation to CD Bus Odd 0			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	CDEVEN1	0x0	RW	**CD Bus Even 1**
	peripheral allocation to CD Bus Even 1			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated
	1	ADC0		The bus is allocated to ADC0
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	CDEVEN0	0x0	RW	**CD Bus Even 0**
	peripheral allocation to CD Bus Even 0			
	Value	Mode		Description
	0	TRISTATE		The bus is not allocated

Bit	Name	Reset	Access	Description
	1	ADC0		The bus is allocated to ADC0

25.6.23 GPIO_EXTIPSELL - External Interrupt Port Select Low

Offset	Bit Position																													
0x400	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x0				0x0				0x0				0x0				0x0				0x0				0x0				0x0	
Access			RW				RW				RW				RW				RW				RW				RW				RW	
Name			EXTIPSEL7				EXTIPSEL6				EXTIPSEL5				EXTIPSEL4				EXTIPSEL3				EXTIPSEL2				EXTIPSEL1				EXTIPSEL0	

Bit	Name	Reset	Access	Description
31:30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29:28	EXTIPSEL7	0x0	RW	**External Interrupt Port Select**

Port select for external interrupt 7 (EXTI7).

Value	Mode	Description
0	PORTA	Port A group selected
1	PORTB	Port B group selected
2	PORTC	Port C group selected
3	PORTD	Port D group selected

27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:24	EXTIPSEL6	0x0	RW	**External Interrupt Port Select**

Port select for external interrupt 6 (EXTI6).

Value	Mode	Description
0	PORTA	Port A group selected
1	PORTB	Port B group selected
2	PORTC	Port C group selected
3	PORTD	Port D group selected

23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	EXTIPSEL5	0x0	RW	**External Interrupt Port Select**

Port select for external interrupt 5 (EXTI5).

Value	Mode	Description
0	PORTA	Port A group selected
1	PORTB	Port B group selected
2	PORTC	Port C group selected
3	PORTD	Port D group selected

Bit	Name	Reset	Access	Description
19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	EXTIPSEL4	0x0	RW	**External Interrupt Port Select**
	Port select for external interrupt 4 (EXTI4).			
	Value	Mode		Description
	0	PORTA		Port A group selected
	1	PORTB		Port B group selected
	2	PORTC		Port C group selected
	3	PORTD		Port D group selected
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	EXTIPSEL3	0x0	RW	**External Interrupt Port Select**
	Port select for external interrupt 3 (EXTI3).			
	Value	Mode		Description
	0	PORTA		Port A group selected
	1	PORTB		Port B group selected
	2	PORTC		Port C group selected
	3	PORTD		Port D group selected
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	EXTIPSEL2	0x0	RW	**External Interrupt Port Select**
	Port select for external interrupt 2 (EXTI2).			
	Value	Mode		Description
	0	PORTA		Port A group selected
	1	PORTB		Port B group selected
	2	PORTC		Port C group selected
	3	PORTD		Port D group selected
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	EXTIPSEL1	0x0	RW	**External Interrupt Port Select**
	Port select for external interrupt 1 (EXTI1).			
	Value	Mode		Description
	0	PORTA		Port A group selected
	1	PORTB		Port B group selected
	2	PORTC		Port C group selected
	3	PORTD		Port D group selected

Bit	Name	Reset	Access	Description
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	EXTIPSEL0	0x0	RW	**External Interrupt Port Select**

Port select for external interrupt 0 (EXTI0).

Value	Mode	Description
0	PORTA	Port A group selected
1	PORTB	Port B group selected
2	PORTC	Port C group selected
3	PORTD	Port D group selected

25.6.24 GPIO_EXTIPSELH - External interrupt Port Select High

Offset	Bit Position																											
0x404	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x0				0x0				0x0				0x0	
Access																			RW				RW				RW				RW	
Name																			EXTIPSEL3				EXTIPSEL2				EXTIPSEL1				EXTIPSEL0	

Bit	Name	Reset	Access	Description
31:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	EXTIPSEL3	0x0	RW	**External Interrupt Port Select**
	Port select for external interrupt 3+8			
	Value	Mode		Description
	0	PORTA		Port A group selected
	1	PORTB		Port B group selected
	2	PORTC		Port C group selected
	3	PORTD		Port D group selected
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	EXTIPSEL2	0x0	RW	**External Interrupt Port Select**
	Port select for external interrupt 2+8			
	Value	Mode		Description
	0	PORTA		Port A group selected
	1	PORTB		Port B group selected
	2	PORTC		Port C group selected
	3	PORTD		Port D group selected
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	EXTIPSEL1	0x0	RW	**External Interrupt Port Select**
	Port select for external interrupt 1+8			
	Value	Mode		Description
	0	PORTA		Port A group selected
	1	PORTB		Port B group selected
	2	PORTC		Port C group selected
	3	PORTD		Port D group selected

Bit	Name	Reset	Access	Description
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	EXTIPSEL0	0x0	RW	**External Interrupt Port Select**
	Port select for external interrupt 0+8			

Value	Mode	Description
0	PORTA	Port A group selected
1	PORTB	Port B group selected
2	PORTC	Port C group selected
3	PORTD	Port D group selected

25.6.25 GPIO_EXTIPINSELL - External Interrupt Pin Select Low

Offset	Bit Position																											
0x408	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x0				0x0				0x0				0x0				0x0				0x0				0x0				0x0	
Access			RW				RW				RW				RW				RW				RW				RW				RW	
Name			EXTIPINSEL7				EXTIPINSEL6				EXTIPINSEL5				EXTIPINSEL4				EXTIPINSEL3				EXTIPINSEL2				EXTIPINSEL1				EXTIPINSEL0	

Bit	Name	Reset	Access	Description
31:30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29:28	EXTIPINSEL7	0x0	RW	**External Interrupt Pin select**

OFFSET select for External Interrupt 7 (EXTI7).

Value	Mode	Description
0	OFFSET0	OFFSET=0
1	OFFSET1	OFFSET=1
2	OFFSET2	OFFSET=2
3	OFFSET3	OFFSET=3

Bit	Name	Reset	Access	Description
27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:24	EXTIPINSEL6	0x0	RW	**External Interrupt Pin select**

OFFSET select for External Interrupt 6 (EXTI6).

Value	Mode	Description
0	OFFSET0	OFFSET=0
1	OFFSET1	OFFSET=1
2	OFFSET2	OFFSET=2
3	OFFSET3	OFFSET=3

Bit	Name	Reset	Access	Description
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	EXTIPINSEL5	0x0	RW	**External Interrupt Pin select**

OFFSET select for External Interrupt 5 (EXTI5).

Value	Mode	Description
0	OFFSET0	OFFSET=0
1	OFFSET1	OFFSET=1
2	OFFSET2	OFFSET=2
3	OFFSET3	OFFSET=3

Bit	Name	Reset	Access	Description
19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	EXTIPINSEL4	0x0	RW	**External Interrupt Pin select**
	OFFSET select for External Interrupt 4 (EXTI4).			
	Value	Mode		Description
	0	OFFSET0		OFFSET=0
	1	OFFSET1		OFFSET=1
	2	OFFSET2		OFFSET=2
	3	OFFSET3		OFFSET=3
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	EXTIPINSEL3	0x0	RW	**External Interrupt Pin select**
	OFFSET select for External Interrupt 3 (EXTI3).			
	Value	Mode		Description
	0	OFFSET0		OFFSET=0
	1	OFFSET1		OFFSET=1
	2	OFFSET2		OFFSET=2
	3	OFFSET3		OFFSET=3
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	EXTIPINSEL2	0x0	RW	**External Interrupt Pin select**
	OFFSET select for External Interrupt 2 (EXTI2).			
	Value	Mode		Description
	0	OFFSET0		OFFSET=0
	1	OFFSET1		OFFSET=1
	2	OFFSET2		OFFSET=2
	3	OFFSET3		OFFSET=3
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	EXTIPINSEL1	0x0	RW	**External Interrupt Pin select**
	OFFSET select for External Interrupt 1 (EXTI1).			
	Value	Mode		Description
	0	OFFSET0		OFFSET=0
	1	OFFSET1		OFFSET=1
	2	OFFSET2		OFFSET=2
	3	OFFSET3		OFFSET=3

Bit	Name	Reset	Access	Description
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	EXTIPINSEL0	0x0	RW	**External Interrupt Pin select**

OFFSET select for External Interrupt 0 (EXTI0).

Value	Mode	Description
0	OFFSET0	OFFSET=0
1	OFFSET1	OFFSET=1
2	OFFSET2	OFFSET=2
3	OFFSET3	OFFSET=3

25.6.26 GPIO_EXTIPINSELH - External Interrupt Pin Select High

Offset	Bit Position																													
0x40C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x0				0x0				0x0				0x0	
Access																			RW				RW				RW				RW	
Name																			EXTIPINSEL3				EXTIPINSEL2				EXTIPINSEL1				EXTIPINSEL0	

Bit	Name	Reset	Access	Description
31:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	EXTIPINSEL3	0x0	RW	**External Interrupt Pin select**

OFFSET select for External Interrupt {b+8} (EXTI{b+8}).

Value	Mode	Description
0	OFFSET8	OFFSET=8
1	OFFSET9	OFFSET=9
2	OFFSET10	OFFSET=10
3	OFFSET11	OFFSET=11

Bit	Name	Reset	Access	Description
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	EXTIPINSEL2	0x0	RW	**External Interrupt Pin select**

OFFSET select for External Interrupt {b+8} (EXTI{b+8}).

Value	Mode	Description
0	OFFSET8	OFFSET=8
1	OFFSET9	OFFSET=9
2	OFFSET10	OFFSET=10
3	OFFSET11	OFFSET=11

Bit	Name	Reset	Access	Description
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	EXTIPINSEL1	0x0	RW	**External Interrupt Pin select**

OFFSET select for External Interrupt {b+8} (EXTI{b+8}).

Value	Mode	Description
0	OFFSET8	OFFSET=8
1	OFFSET9	OFFSET=9
2	OFFSET10	OFFSET=10
3	OFFSET11	OFFSET=11

Bit	Name	Reset	Access	Description
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	EXTIPINSEL0	0x0	RW	**External Interrupt Pin select**
	OFFSET select for External Interrupt {b+8} (EXTI{b+8}).			

Value	Mode	Description
0	OFFSET8	OFFSET=8
1	OFFSET9	OFFSET=9
2	OFFSET10	OFFSET=10
3	OFFSET11	OFFSET=11

25.6.27 GPIO_EXTIRISE - External Interrupt Rising Edge Trigger

Offset	Bit Position																														
0x410	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																					0x0											
Access																					RW											
Name																					EXTIRISE											

Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	EXTIRISE	0x0	RW	**EXT Int Rise**
	External Interrupt n Rising Edge Trigger Enable			

25.6.28 GPIO_EXTIFALL - External Interrupt Falling Edge Trigger

Offset	Bit Position																															
0x414	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																										0x0						
Access																										RW						
Name																										EXTIFALL						

Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	EXTIFALL	0x0	RW	**EXT Int FALL**
	External Interrupt n Falling Edge Trigger Enable			

25.6.29 GPIO_IF - Interrupt Flag

Offset	Bit Position																													
0x420	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0					0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name					EM4WUIF11	EM4WUIF10	EM4WUIF9	EM4WUIF8	EM4WUIF7	EM4WUIF6	EM4WUIF5	EM4WUIF4	EM4WUIF3	EM4WUIF2	EM4WUIF1	EM4WUIF0					EXTIF11	EXTIF10	EXTIF9	EXTIF8	EXTIF7	EXTIF6	EXTIF5	EXTIF4	EXTIF3	EXTIF2	EXTIF1	EXTIF0

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	EM4WUIF11	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
26	EM4WUIF10	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
25	EM4WUIF9	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
24	EM4WUIF8	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
23	EM4WUIF7	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
22	EM4WUIF6	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
21	EM4WUIF5	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
20	EM4WUIF4	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
19	EM4WUIF3	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
18	EM4WUIF2	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
17	EM4WUIF1	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
16	EM4WUIF0	0x0	RW	**EM4 wake up**
	EM4 wakeup interrupt flag			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
11	EXTIF11	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
10	EXTIF10	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
9	EXTIF9	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
8	EXTIF8	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
7	EXTIF7	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
6	EXTIF6	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
5	EXTIF5	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
4	EXTIF4	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
3	EXTIF3	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
2	EXTIF2	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
1	EXTIF1	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			
0	EXTIF0	0x0	RW	**External Pin Flag**
	External Pin interrupt flag			

25.6.30 GPIO_IEN - Interrupt Enable

Offset	Bit Position																													
0x424	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0					0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name					EM4WUIEN11	EM4WUIEN10	EM4WUIEN9	EM4WUIEN8	EM4WUIEN7	EM4WUIEN6	EM4WUIEN5	EM4WUIEN4	EM4WUIEN3	EM4WUIEN2	EM4WUIEN1	EM4WUIEN0					EXTIEN11	EXTIEN10	EXTIEN9	EXTIEN8	EXTIEN7	EXTIEN6	EXTIEN5	EXTIEN4	EXTIEN3	EXTIEN2	EXTIEN1	EXTIEN0

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	EM4WUIEN11	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
26	EM4WUIEN10	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
25	EM4WUIEN9	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
24	EM4WUIEN8	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
23	EM4WUIEN7	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
22	EM4WUIEN6	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
21	EM4WUIEN5	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
20	EM4WUIEN4	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
19	EM4WUIEN3	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
18	EM4WUIEN2	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
17	EM4WUIEN1	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
16	EM4WUIEN0	0x0	RW	**EM4 Wake Up Interrupt En**
	EM4 wakeup interrupt enable			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
11	EXTIEN11	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
10	EXTIEN10	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
9	EXTIEN9	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
8	EXTIEN8	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
7	EXTIEN7	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
6	EXTIEN6	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
5	EXTIEN5	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
4	EXTIEN4	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
3	EXTIEN3	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
2	EXTIEN2	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
1	EXTIEN1	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			
0	EXTIEN0	0x0	RW	**External Pin Enable**
	External Pin interrupt enable			

25.6.31 GPIO_EM4WUEN - EM4 wakeup enable

Offset	Bit Position																													
0x42C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0																											
Access					RW																											
Name					EM4WUEN																											

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:16	EM4WUEN	0x0	RW	**EM4 wake up enable**
	Write 1 to enable EM4 wake up request, write 0 to disable EM4 wake up request			
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

25.6.32 GPIO_EM4WUPOL - EM4 wakeup polarity

Offset	Bit Position																													
0x430	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0																											
Access					RW																											
Name					EM4WUPOL																											

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:16	EM4WUPOL	0x0	RW	**EM4 Wake-Up Polarity**
	EM4 Wakeup Polarity			
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

25.6.33 GPIO_DBGROUTEPEN - Debugger Route Pin enable

Offset	Bit Position																															
0x440	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0x1	0x1	0x1	0x1
Access																													RW	RW	RW	RW
Name																													TDIPEN	TDOPEN	SWDIOTMSPEN	SWCLKTCKPEN

Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	TDIPEN	0x1	RW	**JTAG Test Debug Input Pin Enable**
	Enable JTAG TDI connection to pin.			
2	TDOPEN	0x1	RW	**JTAG Test Debug Output Pin Enable**
	Enable JTAG TDO connection to pin.			
1	SWDIOTMSPEN	0x1	RW	**Route Location 0**
	Enable Serial Wire Data and JTAG Test Mode Select connection to pin. WARNING: When the pin is disabled, the device can no longer be accessed by a debugger. A reset will set the pin back to a default state as enabled. If you disable this pin, make sure you have at least a 3 second timeout at the start of your program code before you disable the pin. This way, the debugger will have time to halt the device after a reset before the pin is disabled.			
0	SWCLKTCKPEN	0x1	RW	**Route Pin Enable**
	Enable Serial Wire and JTAG CLock connection to pin. WARNING: When the pin is disabled, the device can no longer be accessed by a debugger. A reset will set the pin back to a default state as enabled. If you disable this pin, make sure you have at least a 3 second timeout at the start of your program code before you disable the pin. This way, the debugger will have time to halt the device after a reset before the pin is disabled.			

25.6.34 GPIO_TRACEROUTEPEN - Trace Route Pin Enable

Offset	Bit Position																														
0x444	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														RW	RW	RW
Name																														TRACEDATA0PEN	TRACECLKPEN	SWVPEN

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	TRACEDATA0PEN Trace Data0 Pin Enable	0x0	RW	**Trace Data0 Pin Enable**
1	TRACECLKPEN Trace Clk Pin Enable	0x0	RW	**Trace Clk Pin Enable**
0	SWVPEN Serial Wire Viewer Output Pin Enable	0x0	RW	**Serial Wire Viewer Output Pin Enable**

25.6.35 GPIO_CMU_ROUTEEN - CMU pin enable

Offset	Bit Position																														
0x450	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														RW	RW	RW
Name																														CLKOUT2PEN	CLKOUT1PEN	CLKOUT0PEN

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	CLKOUT2PEN	0x0	RW	**CLKOUT2 pin enable control bit**
	CLKOUT2 pin enable control bit			
1	CLKOUT1PEN	0x0	RW	**CLKOUT1 pin enable control bit**
	CLKOUT1 pin enable control bit			
0	CLKOUT0PEN	0x0	RW	**CLKOUT0 pin enable control bit**
	CLKOUT0 pin enable control bit			

25.6.36 GPIO_CMU_CLKIN0ROUTE - CLKIN0 port/pin select

Offset	Bit Position																														
0x454	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CLKIN0 pin select register**
	CLKIN0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CLKIN0 port select register**
	CLKIN0 port select register			

25.6.37 GPIO_CMU_CLKOUT0ROUTE - CLKOUT0 port/pin select

Offset	Bit Position																													
0x458	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CLKOUT0 pin select register**
	CLKOUT0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CLKOUT0 port select register**
	CLKOUT0 port select register			

25.6.38 GPIO_CMU_CLKOUT1ROUTE - CLKOUT1 port/pin select

Offset	Bit Position																													
0x45C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CLKOUT1 pin select register**
	CLKOUT1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CLKOUT1 port select register**
	CLKOUT1 port select register			

25.6.39 GPIO_CMU_CLKOUT2ROUTE - CLKOUT2 port/pin select

Offset	Bit Position																													
0x460	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CLKOUT2 pin select register**
	CLKOUT2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CLKOUT2 port select register**
	CLKOUT2 port select register			

25.6.40 GPIO_DCDC_ROUTEEN - DCDC pin enable

Offset	Bit Position																													
0x46C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																DCDCCOREHIDDENPEN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	DCDCCOREHIDDENPEN	0x0	RW	**DCDCCOREHIDDEN pin enable control bit**
	DCDCCOREHIDDEN pin enable control bit			

25.6.41 GPIO_FRC_ROUTEEN - FRC pin enable

Offset	Bit Position																														
0x47C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																														0x0	0x0	0x0
Access																														RW	RW	RW
Name																														DOUTPEN	DFRAMEPEN	DCLKPEN

Bit	Name	Reset	Access	Description
31:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DOUTPEN	0x0	RW	**DOUT pin enable control bit**
	DOUT pin enable control bit			
1	DFRAMEPEN	0x0	RW	**DFRAME pin enable control bit**
	DFRAME pin enable control bit			
0	DCLKPEN	0x0	RW	**DCLK pin enable control bit**
	DCLK pin enable control bit			

25.6.42 GPIO_FRC_DCLKROUTE - DCLK port/pin select

Offset	Bit Position																														
0x480	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**DCLK pin select register**
	DCLK pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**DCLK port select register**
	DCLK port select register			

25.6.43 GPIO_FRC_DFRAMEROUTE - DFRAME port/pin select

Offset	Bit Position																													
0x484	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**DFRAME pin select register**
	DFRAME pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**DFRAME port select register**
	DFRAME port select register			

25.6.44 GPIO_FRC_DOUTROUTE - DOUT port/pin select

Offset	Bit Position																													
0x488	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**DOUT pin select register**
	DOUT pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**DOUT port select register**
	DOUT port select register			

25.6.45 GPIO_I2C0_ROUTEEN - I2C0 pin enable

Offset	Bit Position																													
0x490	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															SDAPEN	SCLPEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	SDAPEN	0x0	RW	**SDA pin enable control bit**
	SDA pin enable control bit			
0	SCLPEN	0x0	RW	**SCL pin enable control bit**
	SCL pin enable control bit			

25.6.46 GPIO_I2C0_SCLROUTE - SCL port/pin select

Offset	Bit Position																													
0x494	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0x0																	0x0	
Access														RW																	RW	
Name														PIN																	PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**SCL pin select register**
	SCL pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**SCL port select register**
	SCL port select register			

25.6.47 GPIO_I2C0_SDAROUTE - SDA port/pin select

Offset	Bit Position																													
0x498	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**SDA pin select register**
	SDA pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**SDA port select register**
	SDA port select register			

25.6.48 GPIO_I2C1_ROUTEEN - I2C1 pin enable

Offset	Bit Position																													
0x4A0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															SDAPEN	SCLPEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	SDAPEN	0x0	RW	**SDA pin enable control bit**
	SDA pin enable control bit			
0	SCLPEN	0x0	RW	**SCL pin enable control bit**
	SCL pin enable control bit			

25.6.49 GPIO_I2C1_SCLROUTE - SCL port/pin select

Offset	Bit Position																													
0x4A4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN SCL pin select register	0x0	RW	**SCL pin select register**
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT SCL port select register	0x0	RW	**SCL port select register**

25.6.50 GPIO_I2C1_SDAROUTE - SDA port/pin select

Offset	Bit Position																													
0x4A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN SDA pin select register	0x0	RW	**SDA pin select register**
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT SDA port select register	0x0	RW	**SDA port select register**

25.6.51 GPIO_LETIMER0_ROUTEEN - LETIMER pin enable

Offset	Bit Position																													
0x4B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															OUT1PEN	OUT0PEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	OUT1PEN	0x0	RW	**OUT1 pin enable control bit**
	OUT1 pin enable control bit			
0	OUT0PEN	0x0	RW	**OUT0 pin enable control bit**
	OUT0 pin enable control bit			

25.6.52 GPIO_LETIMER0_OUT0ROUTE - OUT0 port/pin select

Offset	Bit Position																													
0x4B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**OUT0 pin select register**
	OUT0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**OUT0 port select register**
	OUT0 port select register			

25.6.53 GPIO_LETIMER0_OUT1ROUTE - OUT1 port/pin select

Offset	Bit Position																													
0x4B8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0x0																	0x0	
Access														RW																	RW	
Name														PIN																	PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**OUT1 pin select register**
	OUT1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**OUT1 port select register**
	OUT1 port select register			

25.6.54 GPIO_EUART0_ROUTEEN - EUART pin enable

Offset	Bit Position																													
0x4C0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0x0	0x0
Access																															RW	RW
Name																															TXPEN	RTSPEN

Bit	Name	Reset	Access	Description
31:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	TXPEN	0x0	RW	**TX pin enable control bit**
	TX pin enable control bit			
0	RTSPEN	0x0	RW	**RTS pin enable control bit**
	RTS pin enable control bit			

25.6.55 GPIO_EUART0_CTSROUTE - CTS port/pin select

Offset	Bit Position																													
0x4C4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN CTS pin select register	0x0	RW	**CTS pin select register**
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT CTS port select register	0x0	RW	**CTS port select register**

25.6.56 GPIO_EUART0_RTSROUTE - RTS port/pin select

Offset	Bit Position																													
0x4C8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN RTS pin select register	0x0	RW	**RTS pin select register**
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT RTS port select register	0x0	RW	**RTS port select register**

25.6.57 GPIO_EUART0_RXROUTE - RX port/pin select

Offset	Bit Position																													
0x4CC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**RX pin select register**
	RX pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**RX port select register**
	RX port select register			

25.6.58 GPIO_EUART0_TXROUTE - TX port/pin select

Offset	Bit Position																													
0x4D0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**TX pin select register**
	TX pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**TX port select register**
	TX port select register			

25.6.59 GPIO_MODEM_ROUTEEN - MODEM pin enable

Offset	Bit Position																													
0x4D8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																		DOUTPEN	DCLKPEN	ANTTRIGSTOPPEN	ANTTRIGPEN	ANTSWUSPEN	ANTSWENPEN	ANTRR5PEN	ANTRR4PEN	ANTRR3PEN	ANTRR2PEN	ANTRR1PEN	ANTRR0PEN	ANTROLLOVERPEN	ANT1PEN	ANT0PEN

Bit	Name	Reset	Access	Description
31:15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	DOUTPEN	0x0	RW	**DOUT pin enable control bit**
	DOUT pin enable control bit			
13	DCLKPEN	0x0	RW	**DCLK pin enable control bit**
	DCLK pin enable control bit			
12	ANTTRIGSTOPPEN	0x0	RW	**ANTTRIGSTOP pin enable control bit**
	ANTTRIGSTOP pin enable control bit			
11	ANTTRIGPEN	0x0	RW	**ANTTRIG pin enable control bit**
	ANTTRIG pin enable control bit			
10	ANTSWUSPEN	0x0	RW	**ANTSWUS pin enable control bit**
	ANTSWUS pin enable control bit			
9	ANTSWENPEN	0x0	RW	**ANTSWEN pin enable control bit**
	ANTSWEN pin enable control bit			
8	ANTRR5PEN	0x0	RW	**ANTRR5 pin enable control bit**
	ANTRR5 pin enable control bit			
7	ANTRR4PEN	0x0	RW	**ANTRR4 pin enable control bit**
	ANTRR4 pin enable control bit			
6	ANTRR3PEN	0x0	RW	**ANTRR3 pin enable control bit**
	ANTRR3 pin enable control bit			
5	ANTRR2PEN	0x0	RW	**ANTRR2 pin enable control bit**
	ANTRR2 pin enable control bit			
4	ANTRR1PEN	0x0	RW	**ANTRR1 pin enable control bit**
	ANTRR1 pin enable control bit			
3	ANTRR0PEN	0x0	RW	**ANTRR0 pin enable control bit**
	ANTRR0 pin enable control bit			

Bit	Name	Reset	Access	Description
2	ANTROLLOVERPEN	0x0	RW	**ANTROLLOVER pin enable control bit**
	ANTROLLOVER pin enable control bit			
1	ANT1PEN	0x0	RW	**ANT1 pin enable control bit**
	ANT1 pin enable control bit			
0	ANT0PEN	0x0	RW	**ANT0 pin enable control bit**
	ANT0 pin enable control bit			

25.6.60 GPIO_MODEM_ANT0ROUTE - ANT0 port/pin select

Offset	Bit Position																													
0x4DC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANT0 pin select register**
	ANT0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANT0 port select register**
	ANT0 port select register			

25.6.61 GPIO_MODEM_ANT1ROUTE - ANT1 port/pin select

Offset	Bit Position																													
0x4E0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANT1 pin select register**
	ANT1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANT1 port select register**
	ANT1 port select register			

25.6.62 GPIO_MODEM_ANTROLLOVERROUTE - ANTROLLOVER port/pin select

Offset	Bit Position																													
0x4E4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTROLLOVER pin select register**
	ANTROLLOVER pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTROLLOVER port select register**
	ANTROLLOVER port select register			

25.6.63 GPIO_MODEM_ANTRR0ROUTE - ANTRR0 port/pin select

Offset	Bit Position																													
0x4E8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTRR0 pin select register**
	ANTRR0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTRR0 port select register**
	ANTRR0 port select register			

25.6.64 GPIO_MODEM_ANTRR1ROUTE - ANTRR1 port/pin select

Offset	Bit Position																													
0x4EC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTRR1 pin select register**
	ANTRR1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTRR1 port select register**
	ANTRR1 port select register			

25.6.65 GPIO_MODEM_ANTRR2ROUTE - ANTRR2 port/pin select

Offset	Bit Position																													
0x4F0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTRR2 pin select register**
	ANTRR2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTRR2 port select register**
	ANTRR2 port select register			

25.6.66 GPIO_MODEM_ANTRR3ROUTE - ANTRR3 port/pin select

Offset	Bit Position																													
0x4F4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTRR3 pin select register**
	ANTRR3 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTRR3 port select register**
	ANTRR3 port select register			

25.6.67 GPIO_MODEM_ANTRR4ROUTE - ANTRR4 port/pin select

Offset	Bit Position																													
0x4F8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTRR4 pin select register**
	ANTRR4 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTRR4 port select register**
	ANTRR4 port select register			

25.6.68 GPIO_MODEM_ANTRR5ROUTE - ANTRR5 port/pin select

Offset	Bit Position																													
0x4FC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTRR5 pin select register**
	ANTRR5 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTRR5 port select register**
	ANTRR5 port select register			

25.6.69 GPIO_MODEM_ANTSWENROUTE - ANTSWEN port/pin select

Offset	Bit Position																													
0x500	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTSWEN pin select register**
	ANTSWEN pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTSWEN port select register**
	ANTSWEN port select register			

25.6.70 GPIO_MODEM_ANTSWUSROUTE - ANTSWUS port/pin select

Offset	Bit Position																													
0x504	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTSWUS pin select register**
	ANTSWUS pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTSWUS port select register**
	ANTSWUS port select register			

25.6.71 GPIO_MODEM_ANTTRIGROUTE - ANTTRIG port/pin select

Offset	Bit Position																													
0x508	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTTRIG pin select register**
	ANTTRIG pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTTRIG port select register**
	ANTTRIG port select register			

25.6.72 GPIO_MODEM_ANTTRIGSTOPROUTE - ANTTRIGSTOP port/pin select

Offset	Bit Position																													
0x50C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ANTTRIGSTOP pin select register**
	ANTTRIGSTOP pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ANTTRIGSTOP port select register**
	ANTTRIGSTOP port select register			

25.6.73 GPIO_MODEM_DCLKROUTE - DCLK port/pin select

Offset	Bit Position																													
0x510	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**DCLK pin select register**
	DCLK pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**DCLK port select register**
	DCLK port select register			

25.6.74 GPIO_MODEM_DINROUTE - DIN port/pin select

Offset	Bit Position																													
0x514	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**DIN pin select register**
	DIN pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**DIN port select register**
	DIN port select register			

25.6.75 GPIO_MODEM_DOUTROUTE - DOUT port/pin select

Offset	Bit Position																													
0x518	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**DOUT pin select register**
	DOUT pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**DOUT port select register**
	DOUT port select register			

25.6.76 GPIO_PDM_ROUTEEN - PDM pin enable

Offset	Bit Position																													
0x520	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																CLKPEN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CLKPEN	0x0	RW	**CLK pin enable control bit**
	CLK pin enable control bit			

25.6.77 GPIO_PDM_CLKROUTE - CLK port/pin select

Offset	Bit Position																													
0x524	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN CLK pin select register	0x0	RW	**CLK pin select register**
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT CLK port select register	0x0	RW	**CLK port select register**

25.6.78 GPIO_PDM_DAT0ROUTE - DAT0 port/pin select

Offset	Bit Position																													
0x528	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN DAT0 pin select register	0x0	RW	**DAT0 pin select register**
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT DAT0 port select register	0x0	RW	**DAT0 port select register**

25.6.79 GPIO_PDM_DAT1ROUTE - DAT1 port/pin select

Offset	Bit Position																													
0x52C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**DAT1 pin select register**
	DAT1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**DAT1 port select register**
	DAT1 port select register			

25.6.80 GPIO_PRS0_ROUTEEN - PRS0 pin enable

Offset	Bit Position																													
0x534	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																	SYNCH3PEN	SYNCH2PEN	SYNCH1PEN	SYNCH0PEN	ASYNCH11PEN	ASYNCH10PEN	ASYNCH9PEN	ASYNCH8PEN	ASYNCH7PEN	ASYNCH6PEN	ASYNCH5PEN	ASYNCH4PEN	ASYNCH3PEN	ASYNCH2PEN	ASYNCH1PEN	ASYNCH0PEN

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	SYNCH3PEN	0x0	RW	**SYNCH3 pin enable control bit**
	SYNCH3 pin enable control bit			
14	SYNCH2PEN	0x0	RW	**SYNCH2 pin enable control bit**
	SYNCH2 pin enable control bit			
13	SYNCH1PEN	0x0	RW	**SYNCH1 pin enable control bit**
	SYNCH1 pin enable control bit			
12	SYNCH0PEN	0x0	RW	**SYNCH0 pin enable control bit**
	SYNCH0 pin enable control bit			
11	ASYNCH11PEN	0x0	RW	**ASYNCH11 pin enable control bit**
	ASYNCH11 pin enable control bit			
10	ASYNCH10PEN	0x0	RW	**ASYNCH10 pin enable control bit**
	ASYNCH10 pin enable control bit			
9	ASYNCH9PEN	0x0	RW	**ASYNCH9 pin enable control bit**
	ASYNCH9 pin enable control bit			
8	ASYNCH8PEN	0x0	RW	**ASYNCH8 pin enable control bit**
	ASYNCH8 pin enable control bit			
7	ASYNCH7PEN	0x0	RW	**ASYNCH7 pin enable control bit**
	ASYNCH7 pin enable control bit			
6	ASYNCH6PEN	0x0	RW	**ASYNCH6 pin enable control bit**
	ASYNCH6 pin enable control bit			
5	ASYNCH5PEN	0x0	RW	**ASYNCH5 pin enable control bit**
	ASYNCH5 pin enable control bit			
4	ASYNCH4PEN	0x0	RW	**ASYNCH4 pin enable control bit**
	ASYNCH4 pin enable control bit			
3	ASYNCH3PEN	0x0	RW	**ASYNCH3 pin enable control bit**

Bit	Name	Reset	Access	Description
	ASYNCH3 pin enable control bit			
2	ASYNCH2PEN	0x0	RW	**ASYNCH2 pin enable control bit**
	ASYNCH2 pin enable control bit			
1	ASYNCH1PEN	0x0	RW	**ASYNCH1 pin enable control bit**
	ASYNCH1 pin enable control bit			
0	ASYNCH0PEN	0x0	RW	**ASYNCH0 pin enable control bit**
	ASYNCH0 pin enable control bit			

25.6.81 GPIO_PRS0_ASYNCH0ROUTE - ASYNCH0 port/pin select

Offset	Bit Position																													
0x538	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH0 pin select register**
	ASYNCH0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH0 port select register**
	ASYNCH0 port select register			

25.6.82 GPIO_PRS0_ASYNCH1ROUTE - ASYNCH1 port/pin select

Offset	Bit Position																													
0x53C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH1 pin select register**
	ASYNCH1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH1 port select register**
	ASYNCH1 port select register			

25.6.83 GPIO_PRS0_ASYNCH2ROUTE - ASYNCH2 port/pin select

Offset	Bit Position																													
0x540	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH2 pin select register**
	ASYNCH2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH2 port select register**
	ASYNCH2 port select register			

25.6.84 GPIO_PRS0_ASYNCH3ROUTE - ASYNCH3 port/pin select

Offset	Bit Position																													
0x544	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH3 pin select register**
	ASYNCH3 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH3 port select register**
	ASYNCH3 port select register			

25.6.85 GPIO_PRS0_ASYNCH4ROUTE - ASYNCH4 port/pin select

Offset	Bit Position																													
0x548	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH4 pin select register**
	ASYNCH4 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH4 port select register**
	ASYNCH4 port select register			

25.6.86 GPIO_PRS0_ASYNCH5ROUTE - ASYNCH5 port/pin select

Offset	Bit Position																													
0x54C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH5 pin select register**
	ASYNCH5 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH5 port select register**
	ASYNCH5 port select register			

25.6.87 GPIO_PRS0_ASYNCH6ROUTE - ASYNCH6 port/pin select

Offset	Bit Position																													
0x550	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH6 pin select register**
	ASYNCH6 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH6 port select register**
	ASYNCH6 port select register			

25.6.88 GPIO_PRS0_ASYNCH7ROUTE - ASYNCH7 port/pin select

Offset	Bit Position																													
0x554	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH7 pin select register**
	ASYNCH7 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH7 port select register**
	ASYNCH7 port select register			

25.6.89 GPIO_PRS0_ASYNCH8ROUTE - ASYNCH8 port/pin select

Offset	Bit Position																													
0x558	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH8 pin select register**
	ASYNCH8 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH8 port select register**
	ASYNCH8 port select register			

25.6.90 GPIO_PRS0_ASYNCH9ROUTE - ASYNCH9 port/pin select

Offset	Bit Position																													
0x55C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH9 pin select register**
	ASYNCH9 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH9 port select register**
	ASYNCH9 port select register			

25.6.91 GPIO_PRS0_ASYNCH10ROUTE - ASYNCH10 port/pin select

Offset	Bit Position																													
0x560	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH10 pin select register**
	ASYNCH10 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH10 port select register**
	ASYNCH10 port select register			

25.6.92 GPIO_PRS0_ASYNCH11ROUTE - ASYNCH11 port/pin select

Offset	Bit Position																													
0x564	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**ASYNCH11 pin select register**
	ASYNCH11 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**ASYNCH11 port select register**
	ASYNCH11 port select register			

25.6.93 GPIO_PRS0_SYNCH0ROUTE - SYNCH0 port/pin select

Offset	Bit Position																													
0x568	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**SYNCH0 pin select register**
	SYNCH0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**SYNCH0 port select register**
	SYNCH0 port select register			

25.6.94 GPIO_PRS0_SYNCH1ROUTE - SYNCH1 port/pin select

Offset	Bit Position																													
0x56C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**SYNCH1 pin select register**
	SYNCH1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**SYNCH1 port select register**
	SYNCH1 port select register			

25.6.95 GPIO_PRS0_SYNCH2ROUTE - SYNCH2 port/pin select

Offset	Bit Position																													
0x570	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**SYNCH2 pin select register**
	SYNCH2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**SYNCH2 port select register**
	SYNCH2 port select register			

25.6.96 GPIO_PRS0_SYNCH3ROUTE - SYNCH3 port/pin select

Offset	Bit Position																													
0x574	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**SYNCH3 pin select register**
	SYNCH3 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**SYNCH3 port select register**
	SYNCH3 port select register			

25.6.97 GPIO_TIMER0_ROUTEEN - TIMER0 pin enable

Offset	Bit Position																													
0x57C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											RW	RW	RW	RW	RW	RW
Name																											CCC2PEN	CCC1PEN	CCC0PEN	CC2PEN	CC1PEN	CC0PEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CCC2PEN	0x0	RW	**CCC2 pin enable control bit**
	CCC2 pin enable control bit			
4	CCC1PEN	0x0	RW	**CCC1 pin enable control bit**
	CCC1 pin enable control bit			
3	CCC0PEN	0x0	RW	**CCC0 pin enable control bit**
	CCC0 pin enable control bit			
2	CC2PEN	0x0	RW	**CC2 pin enable control bit**
	CC2 pin enable control bit			
1	CC1PEN	0x0	RW	**CC1 pin enable control bit**
	CC1 pin enable control bit			
0	CC0PEN	0x0	RW	**CC0 pin enable control bit**
	CC0 pin enable control bit			

25.6.98 GPIO_TIMER0_CC0ROUTE - CC0 port/pin select

Offset	Bit Position																													
0x580	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC0 pin select register**
	CC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC0 port select register**
	CC0 port select register			

25.6.99 GPIO_TIMER0_CC1ROUTE - CC1 port/pin select

Offset	Bit Position																													
0x584	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC1 pin select register**
	CC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC1 port select register**
	CC1 port select register			

25.6.100 GPIO_TIMER0_CC2ROUTE - CC2 port/pin select

Offset	Bit Position																													
0x588	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC2 pin select register**
	CC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC2 port select register**
	CC2 port select register			

25.6.101 GPIO_TIMER0_CCC0ROUTE - CCC0 port/pin select

Offset	Bit Position																													
0x58C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC0 pin select register**
	CCC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC0 port select register**
	CCC0 port select register			

25.6.102 GPIO_TIMER0_CCC1ROUTE - CCC1 port/pin select

Offset	Bit Position																													
0x590	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC1 pin select register**
	CCC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC1 port select register**
	CCC1 port select register			

25.6.103 GPIO_TIMER0_CCC2ROUTE - CCC2 port/pin select

Offset	Bit Position																													
0x594	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC2 pin select register**
	CCC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC2 port select register**
	CCC2 port select register			

25.6.104 GPIO_TIMER1_ROUTEEN - TIMER1 pin enable

Offset	Bit Position																													
0x59C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											RW	RW	RW	RW	RW	RW
Name																											CCC2PEN	CCC1PEN	CCC0PEN	CC2PEN	CC1PEN	CC0PEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CCC2PEN	0x0	RW	**CCC2 pin enable control bit**
	CCC2 pin enable control bit			
4	CCC1PEN	0x0	RW	**CCC1 pin enable control bit**
	CCC1 pin enable control bit			
3	CCC0PEN	0x0	RW	**CCC0 pin enable control bit**
	CCC0 pin enable control bit			
2	CC2PEN	0x0	RW	**CC2 pin enable control bit**
	CC2 pin enable control bit			
1	CC1PEN	0x0	RW	**CC1 pin enable control bit**
	CC1 pin enable control bit			
0	CC0PEN	0x0	RW	**CC0 pin enable control bit**
	CC0 pin enable control bit			

25.6.105 GPIO_TIMER1_CC0ROUTE - CC0 port/pin select

Offset	Bit Position																													
0x5A0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN CC0 pin select register	0x0	RW	**CC0 pin select register**
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT CC0 port select register	0x0	RW	**CC0 port select register**

25.6.106 GPIO_TIMER1_CC1ROUTE - CC1 port/pin select

Offset	Bit Position																													
0x5A4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN CC1 pin select register	0x0	RW	**CC1 pin select register**
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT CC1 port select register	0x0	RW	**CC1 port select register**

25.6.107 GPIO_TIMER1_CC2ROUTE - CC2 port/pin select

Offset	Bit Position																													
0x5A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC2 pin select register**
	CC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC2 port select register**
	CC2 port select register			

25.6.108 GPIO_TIMER1_CCC0ROUTE - CCC0 port/pin select

Offset	Bit Position																													
0x5AC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC0 pin select register**
	CCC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC0 port select register**
	CCC0 port select register			

25.6.109 GPIO_TIMER1_CCC1ROUTE - CCC1 port/pin select

Offset	Bit Position																													
0x5B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC1 pin select register**
	CCC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC1 port select register**
	CCC1 port select register			

25.6.110 GPIO_TIMER1_CCC2ROUTE - CCC2 port/pin select

Offset	Bit Position																													
0x5B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC2 pin select register**
	CCC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC2 port select register**
	CCC2 port select register			

25.6.111 GPIO_TIMER2_ROUTEEN - TIMER2 pin enable

Offset	Bit Position																													
0x5BC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											RW	RW	RW	RW	RW	RW
Name																											CCC2PEN	CCC1PEN	CCC0PEN	CC2PEN	CC1PEN	CC0PEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CCC2PEN	0x0	RW	**CCC2 pin enable control bit**
	CCC2 pin enable control bit			
4	CCC1PEN	0x0	RW	**CCC1 pin enable control bit**
	CCC1 pin enable control bit			
3	CCC0PEN	0x0	RW	**CCC0 pin enable control bit**
	CCC0 pin enable control bit			
2	CC2PEN	0x0	RW	**CC2 pin enable control bit**
	CC2 pin enable control bit			
1	CC1PEN	0x0	RW	**CC1 pin enable control bit**
	CC1 pin enable control bit			
0	CC0PEN	0x0	RW	**CC0 pin enable control bit**
	CC0 pin enable control bit			

25.6.112 GPIO_TIMER2_CC0ROUTE - CC0 port/pin select

Offset	Bit Position																													
0x5C0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC0 pin select register**
	CC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC0 port select register**
	CC0 port select register			

25.6.113 GPIO_TIMER2_CC1ROUTE - CC1 port/pin select

Offset	Bit Position																													
0x5C4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC1 pin select register**
	CC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC1 port select register**
	CC1 port select register			

25.6.114 GPIO_TIMER2_CC2ROUTE - CC2 port/pin select

Offset	Bit Position																													
0x5C8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC2 pin select register**
	CC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC2 port select register**
	CC2 port select register			

25.6.115 GPIO_TIMER2_CCC0ROUTE - CCC0 port/pin select

Offset	Bit Position																													
0x5CC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC0 pin select register**
	CCC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC0 port select register**
	CCC0 port select register			

25.6.116 GPIO_TIMER2_CCC1ROUTE - CCC1 port/pin select

Offset	Bit Position																													
0x5D0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC1 pin select register**
	CCC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC1 port select register**
	CCC1 port select register			

25.6.117 GPIO_TIMER2_CCC2ROUTE - CCC2 port/pin select

Offset	Bit Position																													
0x5D4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC2 pin select register**
	CCC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC2 port select register**
	CCC2 port select register			

25.6.118 GPIO_TIMER3_ROUTEEN - TIMER3 pin enable

Offset	Bit Position																													
0x5DC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											RW	RW	RW	RW	RW	RW
Name																											CCC2PEN	CCC1PEN	CCC0PEN	CC2PEN	CC1PEN	CC0PEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CCC2PEN	0x0	RW	**CCC2 pin enable control bit**
	CCC2 pin enable control bit			
4	CCC1PEN	0x0	RW	**CCC1 pin enable control bit**
	CCC1 pin enable control bit			
3	CCC0PEN	0x0	RW	**CCC0 pin enable control bit**
	CCC0 pin enable control bit			
2	CC2PEN	0x0	RW	**CC2 pin enable control bit**
	CC2 pin enable control bit			
1	CC1PEN	0x0	RW	**CC1 pin enable control bit**
	CC1 pin enable control bit			
0	CC0PEN	0x0	RW	**CC0 pin enable control bit**
	CC0 pin enable control bit			

25.6.119 GPIO_TIMER3_CC0ROUTE - CC0 port/pin select

Offset	Bit Position																													
0x5E0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC0 pin select register**
	CC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC0 port select register**
	CC0 port select register			

25.6.120 GPIO_TIMER3_CC1ROUTE - CC1 port/pin select

Offset	Bit Position																													
0x5E4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC1 pin select register**
	CC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC1 port select register**
	CC1 port select register			

25.6.121 GPIO_TIMER3_CC2ROUTE - CC2 port/pin select

Offset	Bit Position																													
0x5E8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC2 pin select register**
	CC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC2 port select register**
	CC2 port select register			

25.6.122 GPIO_TIMER3_CCC0ROUTE - CCC0 port/pin select

Offset	Bit Position																													
0x5EC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC0 pin select register**
	CCC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC0 port select register**
	CCC0 port select register			

25.6.123 GPIO_TIMER3_CCC1ROUTE - CCC1 port/pin select

Offset	Bit Position																													
0x5F0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC1 pin select register**
	CCC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC1 port select register**
	CCC1 port select register			

25.6.124 GPIO_TIMER3_CCC2ROUTE - CCC2 port/pin select

Offset	Bit Position																													
0x5F4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC2 pin select register**
	CCC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC2 port select register**
	CCC2 port select register			

25.6.125 GPIO_TIMER4_ROUTEEN - TIMER4 pin enable

Offset	Bit Position																													
0x5FC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0	0x0	0x0	0x0	0x0	0x0
Access																											RW	RW	RW	RW	RW	RW
Name																											CCC2PEN	CCC1PEN	CCC0PEN	CC2PEN	CC1PEN	CC0PEN

Bit	Name	Reset	Access	Description
31:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CCC2PEN	0x0	RW	**CCC2 pin enable control bit**
	CCC2 pin enable control bit			
4	CCC1PEN	0x0	RW	**CCC1 pin enable control bit**
	CCC1 pin enable control bit			
3	CCC0PEN	0x0	RW	**CCC0 pin enable control bit**
	CCC0 pin enable control bit			
2	CC2PEN	0x0	RW	**CC2 pin enable control bit**
	CC2 pin enable control bit			
1	CC1PEN	0x0	RW	**CC1 pin enable control bit**
	CC1 pin enable control bit			
0	CC0PEN	0x0	RW	**CC0 pin enable control bit**
	CC0 pin enable control bit			

25.6.126 GPIO_TIMER4_CC0ROUTE - CC0 port/pin select

Offset	Bit Position																													
0x600	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC0 pin select register**
	CC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC0 port select register**
	CC0 port select register			

25.6.127 GPIO_TIMER4_CC1ROUTE - CC1 port/pin select

Offset	Bit Position																													
0x604	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC1 pin select register**
	CC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC1 port select register**
	CC1 port select register			

25.6.128 GPIO_TIMER4_CC2ROUTE - CC2 port/pin select

Offset	Bit Position																													
0x608	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CC2 pin select register**
	CC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CC2 port select register**
	CC2 port select register			

25.6.129 GPIO_TIMER4_CCC0ROUTE - CCC0 port/pin select

Offset	Bit Position																													
0x60C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC0 pin select register**
	CCC0 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC0 port select register**
	CCC0 port select register			

25.6.130 GPIO_TIMER4_CCC1ROUTE - CCC1 port/pin select

Offset	Bit Position																													
0x610	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC1 pin select register**
	CCC1 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC1 port select register**
	CCC1 port select register			

25.6.131 GPIO_TIMER4_CCC2ROUTE - CCC2 port/pin select

Offset	Bit Position																													
0x614	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CCC2 pin select register**
	CCC2 pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CCC2 port select register**
	CCC2 port select register			

25.6.132 GPIO_USART0_ROUTEEN - USART0 pin enable

Offset	Bit Position																													
0x61C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												RW	RW	RW	RW	RW
Name																												TXPEN	CLKPEN	RXPEN	RTSPEN	CSPEN

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	TXPEN	0x0	RW	**TX pin enable control bit**
	TX pin enable control bit			
3	CLKPEN	0x0	RW	**SCLK pin enable control bit**
	SCLK pin enable control bit			
2	RXPEN	0x0	RW	**RX pin enable control bit**
	RX pin enable control bit			
1	RTSPEN	0x0	RW	**RTS pin enable control bit**
	RTS pin enable control bit			
0	CSPEN	0x0	RW	**CS pin enable control bit**
	CS pin enable control bit			

25.6.133 GPIO_USART0_CSROUTE - CS port/pin select

Offset	Bit Position																													
0x620	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CS pin select register**
	CS pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CS port select register**
	CS port select register			

25.6.134 GPIO_USART0_CTSROUTE - CTS port/pin select

Offset	Bit Position																													
0x624	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CTS pin select register**
	CTS pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CTS port select register**
	CTS port select register			

25.6.135 GPIO_USART0_RTSROUTE - RTS port/pin select

Offset	Bit Position																													
0x628	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**RTS pin select register**
	RTS pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**RTS port select register**
	RTS port select register			

25.6.136 GPIO_USART0_RXROUTE - RX port/pin select

Offset	Bit Position																													
0x62C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**RX pin select register**
	RX pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**RX port select register**
	RX port select register			

25.6.137 GPIO_USART0_CLKROUTE - SCLK port/pin select

Offset	Bit Position																													
0x630	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**SCLK pin select register**
	SCLK pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**SCLK port select register**
	SCLK port select register			

25.6.138 GPIO_USART0_TXROUTE - TX port/pin select

Offset	Bit Position																													
0x634	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**TX pin select register**
	TX pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**TX port select register**
	TX port select register			

25.6.139 GPIO_USART1_ROUTEEN - USART1 pin enable

Offset	Bit Position																													
0x63C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												RW	RW	RW	RW	RW
Name																												TXPEN	CLKPEN	RXPEN	RTSPEN	CSPEN

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	TXPEN	0x0	RW	**TX pin enable control bit**
	TX pin enable control bit			
3	CLKPEN	0x0	RW	**SCLK pin enable control bit**
	SCLK pin enable control bit			
2	RXPEN	0x0	RW	**RX pin enable control bit**
	RX pin enable control bit			
1	RTSPEN	0x0	RW	**RTS pin enable control bit**
	RTS pin enable control bit			
0	CSPEN	0x0	RW	**CS pin enable control bit**
	CS pin enable control bit			

25.6.140 GPIO_USART1_CSROUTE - CS port/pin select

Offset	Bit Position																													
0x640	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CS pin select register**
	CS pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CS port select register**
	CS port select register			

25.6.141 GPIO_USART1_CTSROUTE - CTS port/pin select

Offset	Bit Position																													
0x644	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**CTS pin select register**
	CTS pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**CTS port select register**
	CTS port select register			

25.6.142 GPIO_USART1_RTSROUTE - RTS port/pin select

Offset	Bit Position																													
0x648	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**RTS pin select register**
	RTS pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**RTS port select register**
	RTS port select register			

25.6.143 GPIO_USART1_RXROUTE - RX port/pin select

Offset	Bit Position																													
0x64C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**RX pin select register**
	RX pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**RX port select register**
	RX port select register			

25.6.144 GPIO_USART1_CLKROUTE - SCLK port/pin select

Offset	Bit Position																													
0x650	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

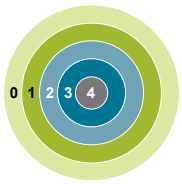
Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**SCLK pin select register**
	SCLK pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**SCLK port select register**
	SCLK port select register			

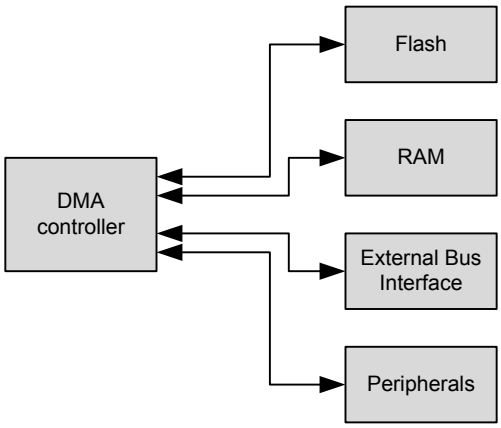
25.6.145 GPIO_USART1_TXROUTE - TX port/pin select

Offset	Bit Position																													
0x654	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0																		0x0	
Access													RW																		RW	
Name													PIN																		PORT	

Bit	Name	Reset	Access	Description
31:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PIN	0x0	RW	**TX pin select register**
	TX pin select register			
15:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PORT	0x0	RW	**TX port select register**
	TX port select register			

26. LDMA - Linked DMA





Quick Facts

What?

The LDMA controller can move data without CPU intervention, effectively reducing the energy consumption for a data transfer.

Why?

The LDMA can perform data transfers more energy efficiently than the CPU and allows autonomous operation in low energy modes.

How?

The LDMA controller has multiple highly configurable, prioritized DMA channels. A linked list of flexible descriptors makes it possible to tailor the controller to the specific needs of an application.

26.1 Introduction

The Linked Direct Memory Access (LDMA) controller performs memory transfer operations independently of the CPU. This has the benefit of reducing the energy consumption and the workload of the CPU, and enables the system to stay in low energy modes while still routing data to memory and peripherals. For example, moving data from the ADC to memory.

26.1.1 Features

- Flexible Source and Destination transfers
  - Memory-to-memory
  - Memory-to-peripheral
  - Peripheral-to-memory
  - Peripheral-to-peripheral
- DMA transfers triggered by peripherals, software, or linked list
- Single or multiple data transfers for each peripheral or software request
- Inter-channel and hardware event synchronization via trigger and wait functions
- Supports single or multiple descriptors
  - Single descriptor
  - Linked list of descriptors
  - Circular and ping-pong buffers
  - Scatter-Gather
  - Looping
  - Pause and restart triggered by other channels
  - Sophisticated flow control which can function without CPU interaction
- Channel arbitration includes:
  - Fixed priority
  - Simple round robin
  - Round robin with programmable multiple interleaved entries for higher priority requesters
- Programmable data size and source and destination address strides
- Programmable interrupt generation at the end of each DMA descriptor execution
- Little-endian/big-endian conversion
- DMA write-immediate function

26.2 Block Diagram

An overview of the LDMA and the modules it interacts with is shown in Figure 26.1 LDMA Block Diagram on page 925.

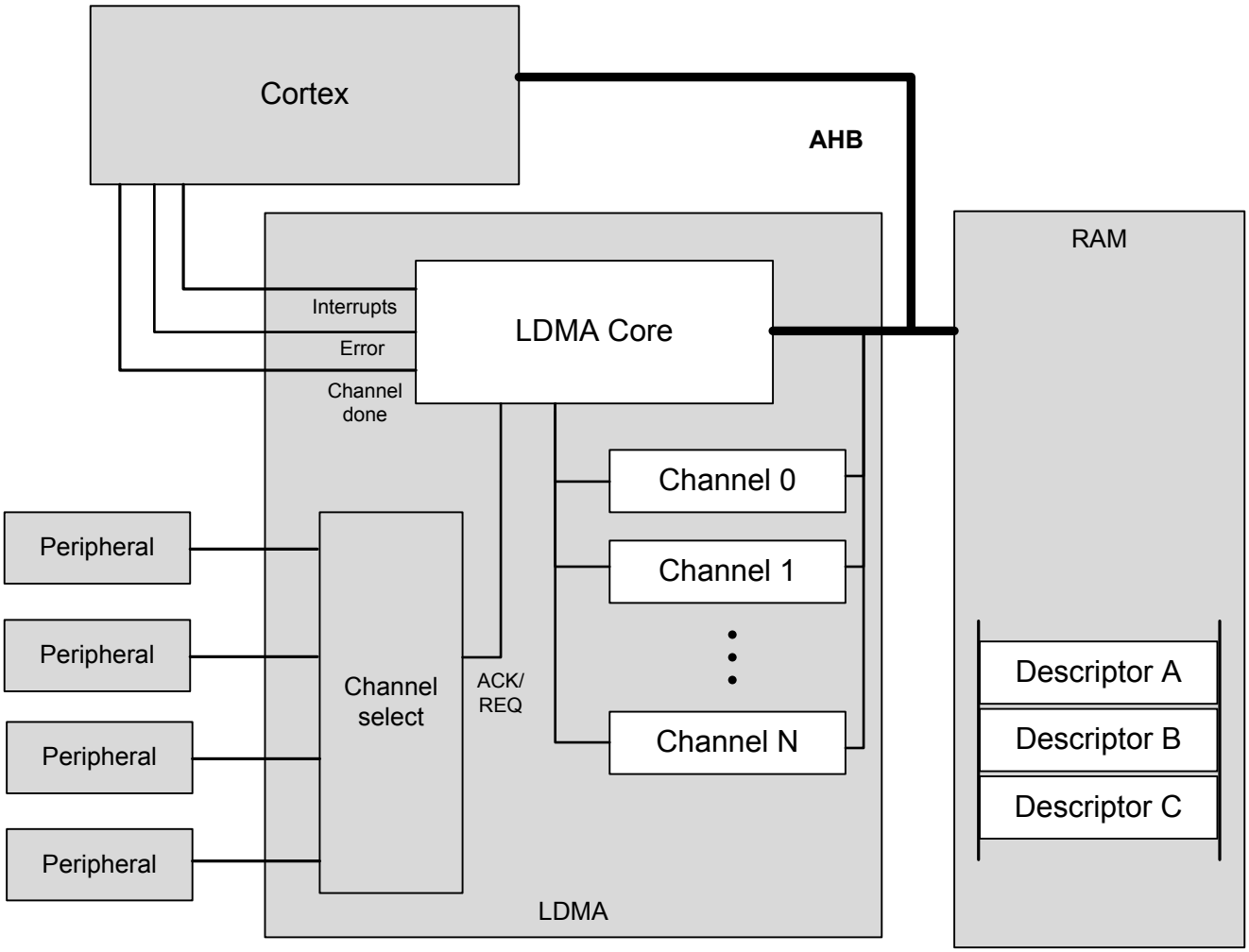


Figure 26.1. LDMA Block Diagram

The Linked DMA Controller consists of three main parts

- A DMA core that executes transfers and communicates status to the core
- A channel select block that routes peripheral DMA requests and acknowledge signals to the DMA
- A set of internal channel configuration registers for tracking the progress of each DMA channel

The DMA has access to all system memory through the AHB bus and the AHB->APB bridge. It can load channel descriptors from memory with no CPU intervention.

26.3 Functional Description

The Linked DMA Controller is highly flexible. It is capable of transferring data between peripherals and memory without involvement from the processor core. This can be used to increase system performance by off-loading the processor from copying large amounts of data or avoiding frequent interrupts to service peripherals needing more data or having available data. It can also be used to reduce the system energy consumption by making the LDMA work autonomously with some EM2/3 peripherals for data transfer without having to wake up the processor core from sleep.

The Linked DMA Controller has 24 independent channels. Each of these channels can be connected to any of the available peripheral DMA transfer request input sources by writing to the channel configuration registers, see 26.3.2 Channel Configuration. In addition, each channel can also be triggered directly by software, which is useful for memory-to-memory transfers.

The channel descriptors determine what the Linked DMA Controller will do when it receives DMA transfer request. The initial descriptor is written directly to the LDMA's channel registers. If desired, the initial descriptor can link to additional linked descriptors stored in memory (RAM or Flash). Alternatively, software may also load the initial descriptor by writing the descriptor address to the LDMA_CHx_LINK register and then setting the corresponding bit the LDMA_LINKLOAD register.

Before enabling a channel, the software must take care to properly configure the channel registers including the link address and any linked descriptors. When a channel is triggered, the Linked DMA Controller will perform the memory transfers as specified by the descriptors. A descriptor contains the memory address to read from, the memory address to write to, link address of the next descriptor, the number of bytes to be transferred, etc. The channel descriptor is described in detail in 26.3.7 Channel descriptor data structure.

The Linked DMA Controller supports both fixed priority and round robin arbitration. The number of fixed and round robin channels is programmable. For round robin channels, the number of arbitration slots requested for each channel is programmable. Using this scheme, it is possible to ensure that timing-critical transfers are serviced on time.

DMA transfers take place by reading a block of data at a time from the source, storing it in the LDMA’s local FIFO, then writing the block out to the destination from the FIFO. Interrupts may optionally be signaled to the CPU’s interrupt controller at the end of any DMA transfer or at the completion of a descriptor if the DONEIFSEN bit is set. An AHB error will always generate an interrupt.

26.3.1 Channel Descriptor

Each DMA channel has descriptor registers. A transfer can be initialized by software writing to the registers or by the DMA itself copying a descriptor from RAM to memory. When using a linked list of descriptors the first descriptor should be initialized by the CPU. The DMA itself will then copy linked descriptors to its descriptor registers as required. In addition to manually initializing the first transfer, software may also cause the LDMA to load the initial descriptor by writing the descriptor address to the LDMA_CHx_LINK register and then setting the corresponding bit the LDMA_LINKLOAD register.

The contents of the descriptor registers are dynamically updated during the DMA transfer. The contents of descriptors in memory are not edited by the controller.

Some descriptor field values are only used for linked descriptors. For example, the SRCMODE and DSTMODE bits of the LDMA_CHx_CTRL registers determine if a linked descriptor is using relative or absolute addressing. Software writes to the address registers will always use absolute addressing and never set these bits. Therefore, these bits are read only.

26.3.1.1 DMA Transfer Size

A DMA transfer is the smallest unit of data that can be transfered by the LDMA. The LDMA supports byte, half-word and word sized transfers. The SIZE field in the LDMA_CHx_CTRL register specifies the data width of one DMA transfer.

26.3.1.2 Source/Destination Increments

The SRCINC and DSTINC in the LDMA_CHx_CTRL register determines the increment between DMA transfers. The increment is in units of DMA transfers and using an increment size of 1 will transfer contiguous bytes, half-words, or words depending on the value of the SIZE field. Multiple unit increments are useful for transferring or packing/unpacking alligned data. For example using an increment of 4 with a size of BYTE will transfer word aligned bytes. An increment of 2 units with a size of HALFWORD is suitable for the transfer of word aligned half-word data. The LDMA can also pack or unpack data by using a different increment size for source and destination. For example - to convert from word aligned byte data (unpacked) to contiguous byte data (packed), set the SIZE to BYTE, SRCINC to 4, and DSTINC to 1.

SIZE may also be set to NONE which will cause the LDMA to read or write the same location for every DMA transfer. This is usefull for accessing peripheral FIFO or data registers.

26.3.1.3 Block Size

The block size defines the amount of data transferred in one arbitration. It consists of one or more DMA transfers. See 26.3.6.1 Arbitration Priority for more details.

26.3.1.4 Transfer Count

The descriptor transfer count defines how many DMA transfers to perform. The number of bytes transferred by the descripter will depend on both the transfer count XFERCNT and the SIZE field settings. TOTAL_BYTES = XFERCNT * SIZE

26.3.1.5 Descriptor List

A descriptor list consists of one or more descriptors which are executed serially. This list may be a simple sequence of descriptors, a loop of descriptors, or a combination of the two.

Each descriptor in the list can be one of several types.

- Single Transfer descriptor: Transfers TOTAL_BYTES of data and then stops.
- Linked Transfer descriptor: Transfers TOTAL_BYTES of data and then loads the next linked descriptor.
- Loop Transfer descriptor: Transfers TOTAL_BYTES of data and performs loop control (see 26.3.2.2 Loop Counter).
- Sync descriptor: Handle synchronization of the list with other entities (see 26.3.7.2 SYNC descriptor structure).
- WRI descriptor: Writes a value to a location in memory (see 26.3.7.3 WRI descriptor structure).

26.3.1.6 Addresses

Before initiating a transfer, software should write the source address, destination address, and if applicable the link address to the descriptor registers. Alternatively, software may load a descriptor from memory by writing the descriptor address to the LDMA_CHx_LINK register and setting the corresponding bit in the LDMA_LINKLOAD register.

During a DMA transfer, the DMA source and destination address registers are pointers to the next transfer address. The LDMA will update the SRC and DST addresses after each transfer. If software halts a DMA transfer by clearing the enable bit, the SRC and DST addresses will indicate the next transfer address.

When a descriptor is finished the DMA will either halt or load the next (linked) descriptor depending on the value of the LINK field in the LDMA_Chx_LINK register. After loading a linked descriptor, the descriptor registers will reflect the content of the loaded descriptor. Note that the linked descriptor must be word aligned in memory. The two least significant bits of the LDMA_CHx_LINK register are used by the LINK and LINKMODE bits. The two least significant bits of the link address are always zero.

26.3.1.7 Addressing Modes

The DMA descriptors support absolute addressing or relative addressing. When using relative addressing, the offset is relative to the current contents of the respective address registers. Regardless of the descriptor addressing modes, the address registers always indicate the absolute address. For example, when loading a descriptor using relative SRC addressing, the LDMA will add the descriptor source address (offset) to the contents of the SRCADDR register (base address). After loading, the SRCADDR register will indicate the absolute address of the loaded descriptor.

The initial descriptor must use absolute addressing. The LDMA will ignore the DSTMODE, SRCMODE, and LINKMODE bits for the initial descriptor and interpret the addresses as an absolute addresses.

Relative addressing is most useful for the link address. The initial descriptor will indicate the absolute address of the linked descriptors in memory. The linked descriptors might be an array of structures. In this case the offset between descriptors is constant and is always 4 words or 16 bytes (each descriptor has 4 words). The LINK address is not incremented or decremented after each transfer. Thus, a relative offset of 0x10 may be used for all linked descriptors.

The source and destination addresses also support relative addressing. When using relative addressing with the source or destination address registers, the LDMA adds the relative offset to the current contents of the respective address register. Since the source and destination addresses are normally incremented after each transfer, the final address will point to one unit past the last transfer. Thus, an offset of zero will give the next sequential data address.

See the example 26.4.6 2D Copy for an common use of relative addressing.

26.3.1.8 Byte Swap

Enabling byte swap reverses the endianness of the incoming source data read into the LDMA's FIFO. Byte swap is only valid for transfer sizes of word and half-word. Note that linked structure reads are not byte swapped.

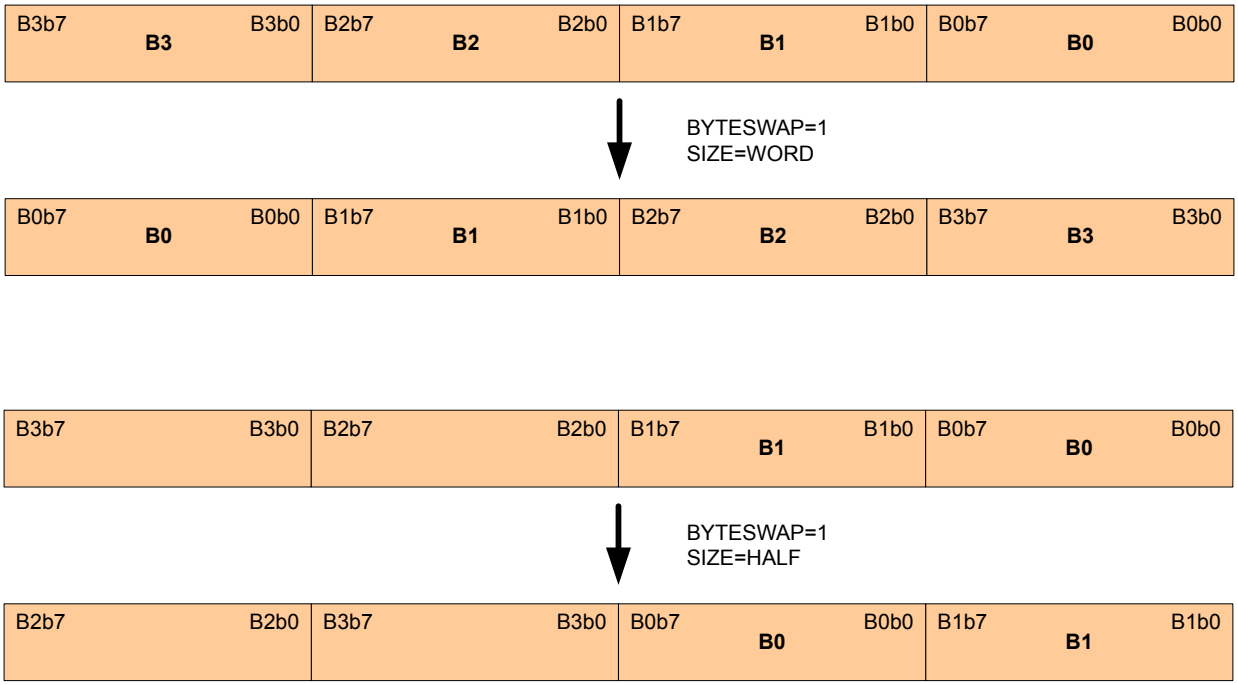


Figure 26.2. Word and Half-Word Endian Byte Swap Examples

26.3.1.9 DMA Size and Source/Destination Increment Programming

The DMA channels' SIZE, SRCINC, and DSTINC bit-fields are programmed to best utilize memory resources. They provide a means for memory packing and unpacking, as well as for matching the size of data being transmitted to or received from an IO peripheral. The following figure shows how 32-bit words of data are read from a memory source into the DMA's internal transfer FIFO, and then written out to the memory destination. The memory organization in bytes is shown as well as the first read to and write from the DMA's FIFO.

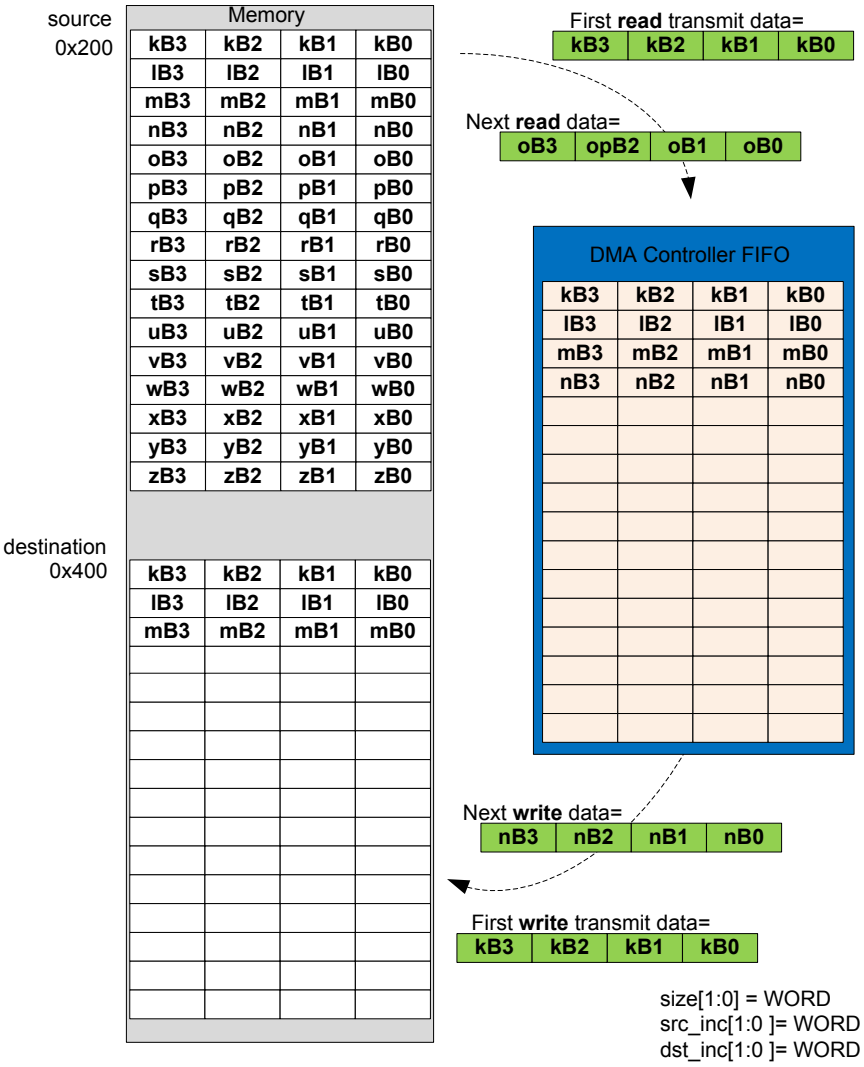


Figure 26.3. Memory-to-Memory Transfer WORD Size Example

The next example shows four variations of half-word sized transfers, with all possible combinations of half- and full-word source and destination increments. Note that when the size and source/destination increments are all configured for half-word, the resulting DMA transfer organization is equivalent to the full-word sized transfer in the previous example. The difference is that the half-word configuration requires twice as many DMA transfers.

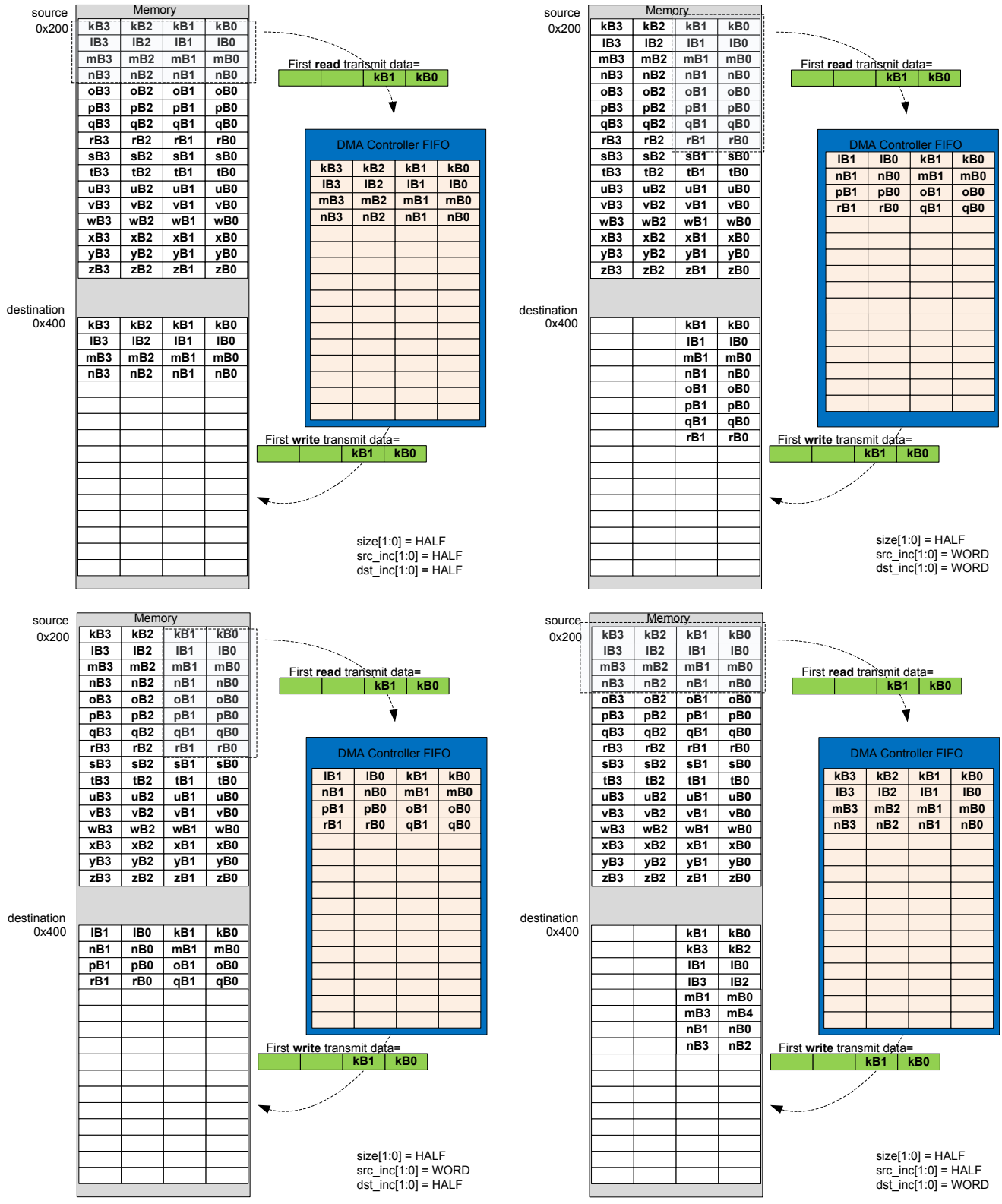


Figure 26.4. Memory-to-Memory Transfer HALF Size Examples

Fields SRCINCSIGN and DSTINCSIGN allow for address decrement. These can be used to mirror an image, for example, in the pixel copy application.

26.3.2 Channel Configuration

Each DMA channel has associated configuration and loop counter registers for controlling direction of address increment , arbitration slots, and descriptor looping.

26.3.2.1 Address Increment/Decrement

Normally DMA transfers increment the source and destination addresses after each DMA transfer. Each channel is also capable of decrementing the source and/or destination addresses after each DMA transfer. This may be useful for flipping an array or copying data from tail to head. For example, a data packet might be prepared as an array of data with increasing addresses and then transmitted from the highest address to the lowest address, from tail to head.

After reset the SRCINCSIGN and DSTINCSIGN bits in the LDMA_CHx_CFG register are cleared causing the source and destination addresses to increment after each transfer. If the SRCINCSIGN bit is set , the DMA will decrement the source address after each transfer. If the DSTINCSIGN bit in the LDMA_CHx_CFG register is set , the DMA will decrement the destination address after each transfer. Setting only one of these bits will flip the data. Setting both bits will copy from tail to head, but will not flip the data.

The SRCINCSIGN and DSTINCSIGN bits apply to all descriptors used by that channel. Software should take care to set the starting source and/or destination address to the highest data address when decrementing.

26.3.2.2 Loop Counter

Each channel has a LDMA_CHx_LOOP register that includes a loop counter field. To use looping, software should initialize the loop counter with the desired number of repetitions before enabling the transfer. A descriptor with the DECLOOPCNT bit set to TRUE will repeat the loop and decrement the loop counter until LOOPCNT = 0.

For a looping descriptor, with DECLOOPCNT=1, the LINK address in the LDMA_CHx_LINK register is used as the loop address. While LOOPCNT is greater than zero, the descriptor will execute and then the LDMA will load the next descriptor using the address specified in the LDMA_CHx_LINK register. This feature enables looping of multiple descriptors. To repeat a single descriptor, the LINK address of the descriptor should point to itself.

After LOOPCNT reaches zero, if the LINK bit in the descriptor LINK word is clear the transfer stops. If the LINK bit is set, the LDMA will load the next sequential descriptor located immediately following the looping descriptor. The behavior of the LINK bit is different for a looping descriptor. This is necessary because the LINK address is re-purposed as the loop address for a looping descriptor.

Note that LOOPCNT sets the number of repeats, not the number of iterations. The total number of loop iterations will be LOOPCNT plus 1. Normally, the LOOPCNT should be set to one or more repeats.

Also note that because there is only one LOOPCNT per channel, software intervention is required to update the LOOPCNT if a sequence of transfers contains multiple loops. It is also possible to use a write immediate DMA data transfer to update the LDMA_CHx_LOOP register.

26.3.3 Channel Select Configuration

The channel select block determines which peripheral request signal connects to each DMA channel.

This configuration is done by software through the SOURCESEL and SIGSEL fields of the LDMA_CHn_REQSEL register. SOURCESEL selects the peripheral and SIGSEL picks which DMA request signals to use from the selected peripheral. Please refer to 26.5 LDMA Source Selection Details for more information.

26.3.4 Starting a transfer

A transfer may be started by software, a peripheral request, or a descriptor load.

Software may initiate a transfer by setting the bit for the desired channel in the LDMA_SREQ register. In this case the channel should set SOURCESEL to NONE to prevent unintentional triggering of the channel by a peripheral.

A peripheral may trigger the channel by configuring the peripheral source and signal as described in 26.3.3 Channel Select Configuration

The LDMA may also be configured to begin a transfer immediately after a new descriptor is loaded by setting the STRUCTREQ field of the LDMA_CHx_CTRL register or descriptor word.

This configuration is done by software through the SOURCESEL and SIGSEL fields of the LDMA_CHn_REQSEL register. SOURCESEL selects the peripheral and SIGSEL picks which DMA request signals to use from the selected peripheral.

26.3.4.1 Peripheral Transfer Requests

By default peripherals issue a Single Request (SREQ) when any data is present. For peripherals with a data buffer or FIFO this occurs any time the FIFO is not empty. Upon receiving an SREQ the LDMA will perform one DMA transfer and stop till another request is made.

It is generally more efficient to wait for a peripheral to accumulate data and transfer in a burst. This both reduces overhead of the DMA engine and allows EM2 peripherals to save power by using the LDMA less often. To enable this set the IGNORESREQ bit in the LDMA_CHx_CTRL register (or descriptor) which will cause the LDMA to ignore SREQ's and wait for a full Request (REQ) signal. When the REQ is received the entire descriptor will be executed. For most peripherals with a FIFO the REQ signal is set when the FIFO is full, or a predetermined threshold has been reached. See the individual peripheral chapters for more information.

26.3.5 Managing Transfer Errors

LDMA transfer errors are normally managed using interrupts. Software should clear the ERROR flag in the bit in the LDMA_IF register and enable error interrupts by setting the ERROR bit in the LDMA_IEN register before initiating a DMA transfer.

The LDMA interrupt handler should check the ERROR flag bit in the LDMA_IF register. If the ERROR flag bit is set, it should then read the CHERROR field in the LDMA_STATUS register to determine the errant channel. The interrupt handler should reset the channel and clear the ERROR flag bit in the LDMA_IF register before returning.

26.3.6 Arbitration

While multiple channels are configured simultaneously the LDMA engine can only be actively copying data for one channel at a time. Arbitration determines which channel is being serviced at any point in time. The LDMA will choose a channel through arbitration, transfer BLOCK_SIZE elements of that channel and then arbitrate again choosing another channel to service. This allows high priority channels to be serviced while lower priority channels are in the middle of a transfer.

26.3.6.1 Arbitration Priority

There are two modes in determining priority when the controller arbitrates: fixed priority and round robin priority.

In fixed priority mode, channel 0 has the highest priority. As the channel number increases, the priority decreases. When the LDMA controller is idle or when a transfer completes, the highest priority channel with an active request is granted the transfer. This mode guarantees smallest latency for the highest priority requesters. It is best suited for systems where peak bandwidth is well below LDMA controller’s maximum ability to serve. The drawback of this mode is the possibility of starvation for lowest priority requesters.

In the round robin priority mode, each active requesting channel is serviced in the order of priority. A late arriving request on a higher priority channel will not get serviced until the next round. This mode minimizes the risk of starving low-priority latency-tolerant requesters. The drawback of this mode is higher risk of starving low-latency requesters.

The NUMFIXED field in the LDMA_CTRL register determines which channels are fixed priority and which are round robin. Channels lower than NUMFIXED are fixed priority while those above it are round robin. A value of 0x0 implies all channels are round robin. A value of 0x4 implies channels 0 through 3 are fixed priority and 4 through 7 are round robin. A value of 7 implies that channels 0 through 6 are fixed and channel 7 is round robin. This is functionally equivalent to having 8 fixed priority channels.

Fixed priority channels always take priority over round robin. As long as NUMFIXED is greater than 0, there is a possibility that a higher priority channel can starve the remaining channels.

To address the drawbacks of using fixed priority or round robin priority the LDMA implements the concept of arbitration slots. This allows for channels to have high bandwidth and low latency while preventing starvation of latency tolerant low priority channels.

Each channel has a two bit ARBSLOT field in its LDM_CHx_CFG register. This field only applies to channels marked as round robin (determined by NUMFIXED). The channels in the same arbitration slot are treated equally with round robin scheduling. Channels marked with a higher arbitration slot will get serviced more frequently. By default all channels are placed in arbitration slot 1.

Every time the channels in slot 1 get serviced the channels in slot 2 get serviced twice, those in slot 4 get serviced 4 times, and those in slot 8 get serviced 7 times. The specific arbitration allocation can be seen by the following table. The highest arbitration slot is serviced every other arbitration cycle, allowing for low latency response. If there are no requests from channels in arbitration slot then that slot is immediately skipped.

Table 26.1. Arbitration Slot Order

Arbslot order	8	4	8	2	8	4	8	1	8	4	8	2	8	4
Arbslot1								1						
Arbslot2				1								1		
Arbslot4		1				1				1				1
Arbslot8	1		1		1		1		1		1		1	

The top row shows the order at which the arbitration slots are executed. The remaining part of the table shows a more visual interpretation of the arbitration order.

For example, if we have one low latency channel (CHNL0) and two latency tolerant channels (CHNL1 and CHNL2). We could use the following settings.

LDMA_CTRL.NUMFIXED = 0; set round robin for all channels.

CHNL0_CFG.ARBSLOTS = TWO;

CHNL1_CFG.ARBSLOTS = ONE;

CHNL2_CFG.ARBSLOTS = ONE;

If all channels are constantly requesting transfers, then the arbitration order is: CHNL0, CHNL1, CHNL0, CHNL2, CHNL0, CHNL1, CHNL0, CHNL2, CHNL0, etc

Note, there are no channels assigned to arbitration slot four or eight in this example, so those slots are skipped and the final sequence is ARBSLOT2, ARBSLOT1, ARBSLOT2, ARBSLOT1, etc...

Channel 1 and Channel 2 are selected in round robin order when arbitration slot 1 is executed.

If we replace the ARBSLOTS value for channel 0 with EIGHT, then the sequence would look like the following:

CHNL0, CHNL0, CHNL0, CHNL0, CHNL1, CHNL0, CHNL0, CHNL0, CHNL2, CHNL0, CHNL0, CHNL0, CHNL0, CHNL1, etc.

26.3.6.2 DMA Transfer Arbitration

In addition to the inter channel arbitration, software can configure when the controller arbitrates during a DMA transfer. This provides reduced latency to higher priority channels when configuring low priority transfers with more arbitration cycles.

The LDMA provides four bits that configure how many DMA transfers occur before it re-arbitrates. These bits are known as the BLOCKSIZE bits and they map to the arbitration rate as shown below. For example, if BLOCKSIZE = 4 then the arbitration rate is 6, that is, the controller arbitrates every 6 DMA transfers.

Table 26.2 AHB bus transfer arbitration interval on page 934 lists the arbitration rates.

Table 26.2. AHB bus transfer arbitration interval

BLOCKSIZE	Arbitrate After x DMA transfers
0	x = 1
1	x = 2
2	x = 3
3	x = 4
4	x = 6
5	x = 8
6	x = 12
7	x = 16
8	x = 24
9	x = 32
10	x = 64
11	x = 128
12	x = 256
13	x = 512
14	x = 1024
15	x = lock

Note: Software must take care not to assign a low-priority channel with a large BLOCKSIZE because this prevents the controller from servicing high-priority requests, until it re-arbitrates.

The number of DMA transfers that need to be done is specified by the user in XFERCNT. When XFERCNT > BLOCKSIZE and is not an integer multiple of BLOCKSIZE then the controller always performs sequences of BLOCKSIZE transfers until XFERCNT < BLOCKSIZE remain to be transferred. The controller performs the remaining XFERCNT transfers at the end of the DMA cycle.

Software must store the value of the BLOCKSIZE bits in the channel control data structure. See 26.3.7.1 XFER descriptor structure for more information about the location of the BLOCKSIZE bits in the data structure.

26.3.7 Channel descriptor data structure

Each channel descriptor consists of four 32-bit words:

- CTRL - control word contains information like transfer count and block size.
- SRC - source address points to where to copy data from
- DST - destination address points to where to copy data to
- LINK - link address points to where to load the next linked descriptor

These words map directly to the LDMA_CHx_CTRL, LDMA_CHx_SRC, LDMA_CHx_DST, and LDMA_CHx_LINK registers. The usage of the SRC and DST fields may differ depending on the structure type

There are three different types of descriptor data structures: **XFER, SYNC, and WRI**

26.3.7.1 XFER descriptor structure

This descriptor defines a typical data transfer which may be a Normal, Link, or Loop transfer.

Only this structure type can be written directly into LDMA's registers by the CPU. All descriptors may be linked to. Please refer to the register descriptions for additional information.

For specifying XFER structure type, set STRUCTTYPE to 0. Please see the peripheral register descriptions for information on the fields in this structure.

Name	Bit Position																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRL	DSTMODE	SRCMODE	DSTINC		SIZE		SRCINC		IGNORESREQ	DECLOOPCNT	REQMODE	DONEIFSEN	BLOCKSIZE				BYTESWAP	XFERCNT											STRUCTREQ		STRUCTTYPE	
SRC	SRCADDR																															
DST	DSTADDR																															
LINK	LINKADDR																														LINK	LINKMODE

26.3.7.2 SYNC descriptor structure

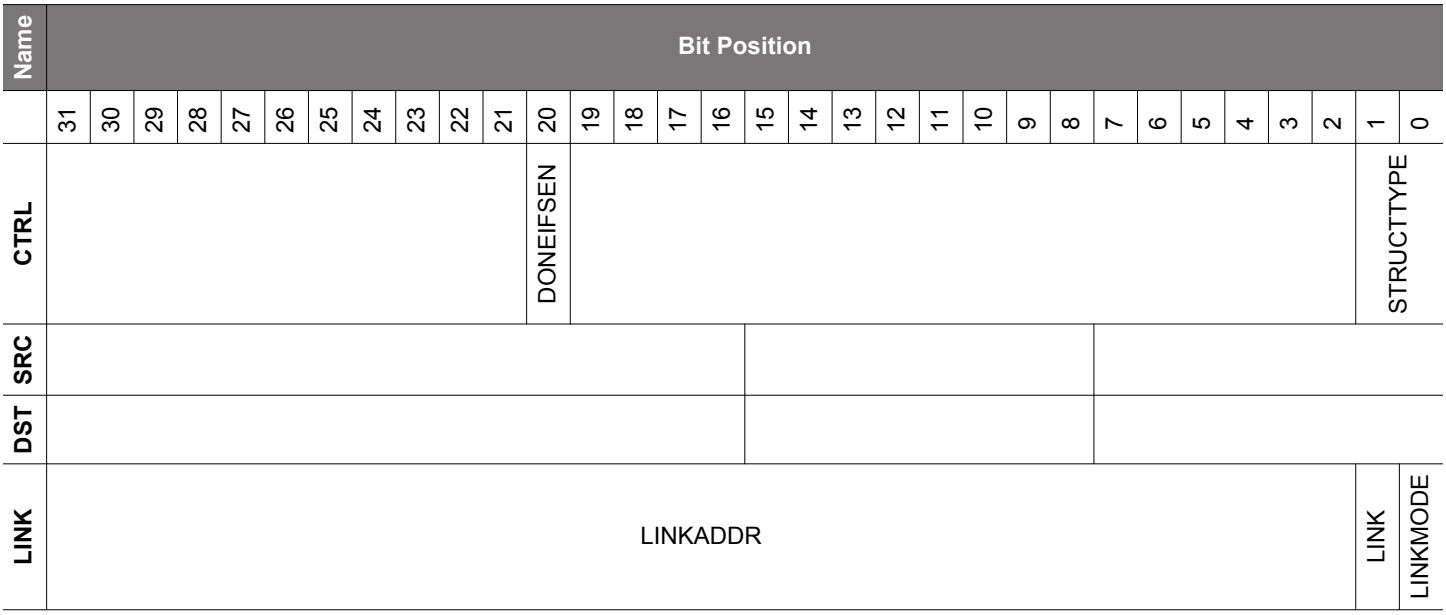
This descriptor defines an intra-channel synchronizing structure. It allows the channel to wait for some external stimulus before continuing on to the next descriptor. This structure is also used to provide stimulus to another channel to indicate that it may continue.

For example channel 1 may be configured to transfer a header into a buffer while channel 2 is simultaneously transferring data into the same structure. When channel 1 has completed it can wait for a sync signal from channel 2 before transferring the now complete buffer to a peripheral.

Synch descriptors do nothing until a condition is met. The condition is formed by the SYNCTRIG field in the LDMA_SYNC register and the MATCHEN and MATCHVAL fields of the descriptor. When (SYNCTRIG & MATCHEN) == (MATCHVAL & MATCHEN) the next descriptor is loaded. In addition to waiting for the condition a Link descriptor can set or clear bits in SYNCTRIG to meet the conditions of another channel and cause it to continue. The CPU also has the ability to set and clear the SYNCTRIG bits from software.

This structure type can only be linked in from memory.

For specifying SYNC structure type, set STRUCTTYPE to 1.



Bit	Name	Description
1:0	STRUCTTYPE	**Descriptor Type**
	This field indicates which type of descriptor this is. It must be 1 for a SYNC descriptor.	
20	DONEIFSEN	**Done if Set indicator**
	If set the interrupt flag will be set when descriptor completes.	
15:8	SYNCCLR	**Sync Trigger Clear**
	This bit-field is used to clear corresponding bits within the SYNCTRIG field of the SYNC LDMA_SYNC register. To clear a given bit, a one should be loaded to the corresponding bit. Set is given priority over clear if both corresponding bits are loaded with a one. The sync trigger clear function can only be used when loaded from a linked structure. Alternately, the user can directly write the SYNCTRIG bit-field if required.	
7:0	SYNCSET	**Sync Trigger Set**
	This bit-field is used to set corresponding bits within the SYNCTRIG bit-field. To set a given bit, a one should be loaded to the corresponding bit. Set is given priority over clear if both corresponding bits are loaded with a one. The sync trigger set function can only be used when loaded from a linked structure. Alternately, the user can directly write the SYNCTRIG bit-field if required.	
15:8	MATCHEN	**Sync Trigger Match Enable**
	This bit-field serves as the SYNCTRIG match enable. A sync match triggers the load of the next linked DMA structure as specified by link_mode, when: (SYNCTRIG & MATCHEN) == (MATCHVAL & MATCHEN).	
7:0	MATCHVAL	**Sync Trigger Match Value**

Bit	Name	Description
		This bit-field serves as the SYNCTRIG match value. A sync match triggers the load of the next linked DMA structure as specified by link_mode, when: (SYNCTRIG & MATCHEN) == (MATCHVAL & MATCHEN).

26.3.7.3 WRI descriptor structure

This descriptor defines a write-immediate structure. This allows a list of descriptors to write a value to a register or memory location. For example, if a channel wishes to perform two loops in a descriptor sequence a WRI may be used to program the loop count for the second loop.

This structure type can only be linked in from memory.

For specifying WRI structure type, set STRUCTTYPE to 2.

Name	Bit Position																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRL												DONEIFSEN																			STRUCTTYPE	
SRC	SRC																															
DST	DSTADDR																															
LINK	LINKADDR																														LINK	LINKMODE

Bit	Name	Description
1:0	STRUCTTYPE	**Descriptor Type**
		This field indicates which type of descriptor this is. It must be 2 for a WRI descriptor.
20	DONEIFSEN	**Done if Set indicator**
		If set the interrupt flag will be set when descriptor completes.
31:0	IMMVAL	**Immediate Value for Write**
		This bit-field specifies the immediate data value that is to be written to the address pointed to by DSTADDR. Only one write occurs for WRI structures.
31:0	DSTADDR	**Address to write**
		This bit-field specifies the address the immediate data should be written to.

26.3.8 Interaction with the EMU

The LDMA interacts with the Energy Management Unit (EMU) to allow transfers from a low energy peripheral while in EM2.

When using the ADC in EM2 or EM3 the EMU can wake up the LDMA as needed to allow data transfers to occur.

26.3.9 Interrupts

The LDMA_IF Interrupt flag register contains one DONE bit for each channel and one combined ERROR bit. When enabled, these interrupts are available as interrupts to the M33 core. They are combined into one interrupt vector, DMA_INT. If the interrupt for the DMA is enabled in the ARM M33 core, an interrupt will be made if one or more of the interrupt flags in LDMA_IF and their corresponding bits in LDMA_IEN are set.

When a descriptor finishes execution the interrupt flag for that channel will be set if the DONEIFSEN field of the LDMA_CHx_LOOP register is set. If LINK and DONEIFSEN are both set when the descriptor completes the interrupt and the linked descriptor will be immediately loaded. When the final descriptor in a linked list (LINK = 0) is finished the interrupt flag is always set regardless of the state of DONEIFSEN.

26.3.10 Debugging

For a peripheral request DMA transfer, if software sets a bit for a channel in the LDMA_DBGHALT register then the DMA will halt during a debug halt and the SRC and DST registers in the debug window will show the transfer in progress. Otherwise, during debug halt the DMA will continue to run and complete the entire transfer causing the descriptor registers to indicate the transfer has completed.

26.4 Examples

This section provides examples of common LDMA usage. All examples assume the LDMA is in the reset state with the channel being configured disabled and LDAM_CHx_CFG, LDMA_CHx_LOOP, and LDMA_CHx_LINK cleared.

26.4.1 Single Direct Register DMA Transfer

This simple example uses only the Channel Descriptor registers directly and does not use linking. Software writes directly to the LDMA channel registers. This example does not use a memory based descriptor list.

This example is suitable for most simple transfers that are limited to transferring one block of data. It supports anything that can be done using a single descriptor. This includes endian conversion and packing/unpacking data. Channel 0 is used for this example.

The LDMA will be used to copy 127 contiguous half words (254 bytes) from 0x0 to 0x1000. It will allow arbitration every 4 transfers and is triggered by a CPU write to the LDMA_SWREQ register. The CH0 interrupt flag will be set when the transfer completes since the descriptor does not link to another descriptor.

- Configure LDMA_CH0_CTRL
  - DSTMODE = 0 (absolute)
  - SRCMODE = 0 (absolute)
  - SIZE = HALFWORD (16 bits)
  - DSTINC = 0 (1 half-word)
  - SRCINC = 0 (1 half-word)
  - DECLOOPCNT=0 (unused)
  - REQMODE = 1 (one request transfers all data)
  - BLOCKSIZE = 3 (4 transfers)
  - BYTESWAP=0 (no byte swap)
  - XFERCNT=127 (transfer 127 half words)
  - STRUCTTPYE=0 (TRANSFER)
- Write source address to LDMA_CH0_SRC register
- Write destination address to LDMA_CH0_DST register
- Configure the LDMA_CH0REQSEL register for the desired peripheral or select none for a memory-to-memory transfer
- Clear and enable interrupts.
  - Write a 1 to bit 0 of the LDMA_IFC register to clear the CH0 DONE flag
  - Write a 1 to bit 0 of the LDMA_IEN register to enable the CH0 interrupt
- Write a 1 to bit 0 of the LDMA_CHEN register to enable CH0

The REQMODE field is normally cleared to zero for a peripheral request transfer and will transfer the specified block size for each peripheral request. The REQMODE may be set to 1 for a memory-to-memory transfer or any time it is desired for a single DMA request to initiate complete transfer.

26.4.2 Descriptor Linked List

This example shows how to use a Linked List of descriptors. Each descriptor has a link address which points to the next descriptor in the list. A descriptor may be removed from the Linked list by altering the Link address of the one before it to point to the one after it. Descriptor Linked lists are useful when handling an array of buffers for communication data. For example, a bad packet can be removed from a receiver queue by simply removing the descriptor from the linked list.

Software loads the first descriptor into the DMA by writing the descriptor address to LDMA_CHx_LINK and setting the bit for that channel in the LDMA_LINKLOAD register. This method is preferred when using a linked list in memory since it treats the first descriptor just like all the others. However, it is also allowed for software to write the first descriptor directly to the LDMA registers.

In this example 4 descriptors are executed in series. the interrupt flag is set after the 2nd and 4th (last) descriptors have completed.

- Prepare a list of descriptors using the XFER structure type in RAM
- Initialize the CTRL, SRC, and DST members as desired
  - Setting STRUCTREQ in the CTRL word for descriptors 2-4 will cause them to begin transferring data as soon as they are loaded.
- Write 0x00000013 to the LINK member of all but the last descriptor
  - LINKMODE = 1 (relative addressing)
  - LINK = 1 (Link to the next descriptor)
  - LINKADDR = 0x00000010 (size of descriptor)
- Set the DONEIFSEN bit in the CTRL member of the 2nd structure so that the interrupt flag will be set when it completes
- Write 0x00000000 to the LINK member of the last descriptor
  - LINK = 0 (Do not link to the next descriptor)
  - LINKMODE = 0 (don't care)
  - LINKADDR = 0x00000000 (don't care)

Each descriptor now points to the start of the next descriptor as shown on the left in Figure 26.5 Descriptor Linked List on page 939. To remove a descriptor from the linked list modify the LINK address of the descriptor of the one before to point to the one after. For example to remove the third descriptor, add 0x00000010 to the LINK register of the second descriptor. The second descriptor will now point to the forth descriptor and skip over the third descriptor as shown on the right in Figure 26.5 Descriptor Linked List on page 939.

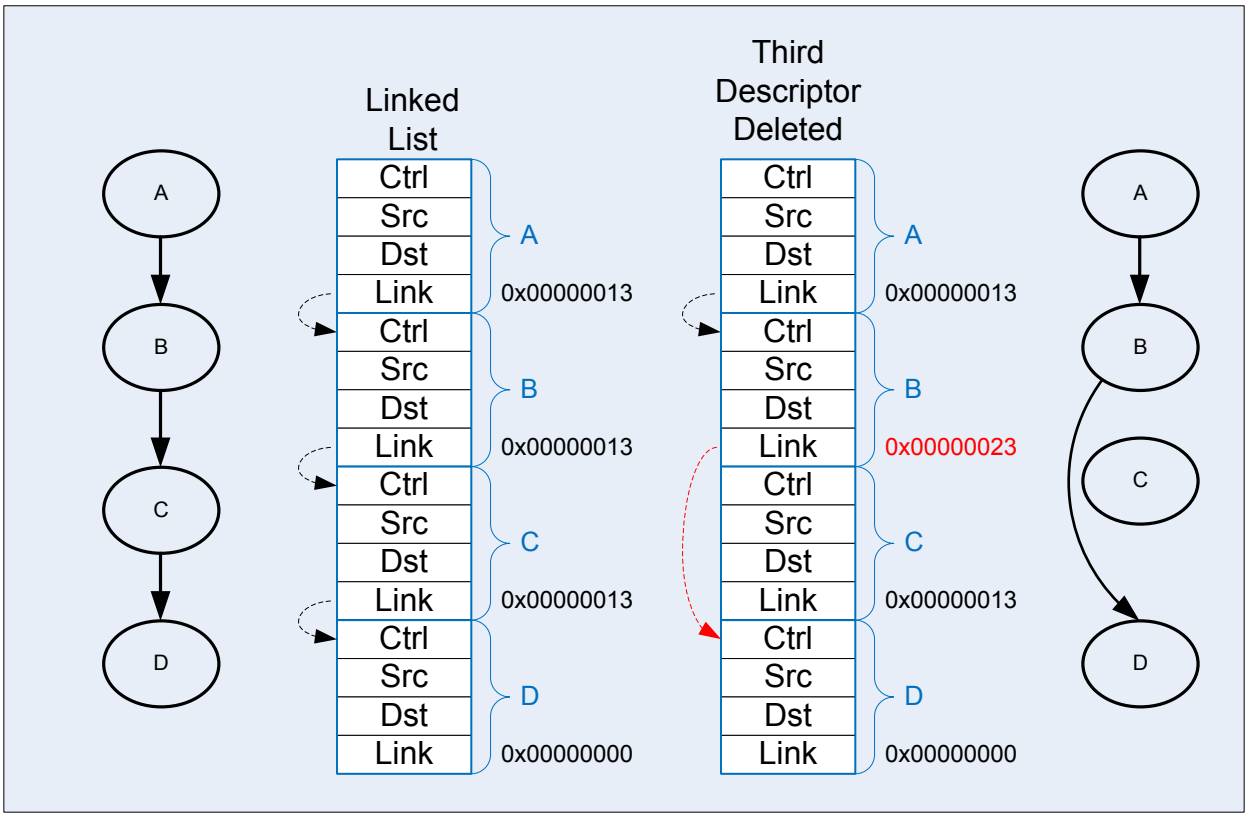


Figure 26.5. Descriptor Linked List

To start execution of the linked list of descriptors:

- Write the absolute address of the first descriptor to the LINKADR field of the LDMA_CH0_LINK register
- Set the LINK bit of LDMA_CH0_LINK register.
- Configure the LDMA_CH0REQSEL register for the desired peripheral or select none for memory-to-memory
- Clear and enable interrupts as desired
- Set bit 0 in the LDMA_LINKLOAD register to initiate loading and execution of the first descriptor

Alternatively, software can manually copy the first descriptor contents to the LDMA_CH0_CTRL, LDMA_CH0_SRC, LDMA_CH0_DST, and LDMA_CH0_LINK registers and then enable the channel in the LDMA_CHEN register.

26.4.3 Single Descriptor Looped Transfer

This example demonstrates how to use looping using a single descriptor. This method allows a single DMA transfer to be repeated a specified number of times. The looping descriptor is stored in memory and reloaded by hardware. After a specified number of iterations, the transfer stops.

CH0 is setup to copy 4 words from the ADC FIFO into a 15 word buffer at 0x1000. It repeats 4 times to fill the entire 16 word buffer. An interrupt will fire when the entire 16 words have been transferred.

Initialize the Linked descriptor in memory as follows:

- Configure CTRL member
  - DSTMODE = 0 (absolute)
  - SRCMODE = 0 (absolute)
  - SIZE = WORD
  - DSTINC = 0 (1 WORD)
  - SRCINC = 3 (0 WORDS)
  - DECLOOPCNT=1 (decrement loop count)
  - REQMODE=1 (Use XFERCNT)
  - BLOCKSIZE = 4 (4 words)
  - BYTESWAP=0 (no swap)
  - XFERCNT= 4 (4 words)
  - STRUCTTPYE=0 (TRANSFER)
  - IGNORESREQ=1 (ignore single requests)
- Write the address ADC0_SINGLEDATA register to the SRC member
- Write 0x1000 address to DST member
- Configure the LINK member
  - LINK = 0 (stop after loop)
  - MODE = 1 (relative link address)
  - LINKADDR = 0 (point to ourself)
- Configure the Channel
  - Write the desired number of repeats to the LDMA_CH0_LOOP register
  - SOURCESEL in LDMA_CH0REQSEL = ADC0 (select the ADC)
  - SIG in LDMA_CH0REQSEL = ADC0SCAN (select the scan conversion request)
- Clear and enable interrupts
- Load the descriptor using LINKLOAD as described in 26.4.2 Descriptor Linked List

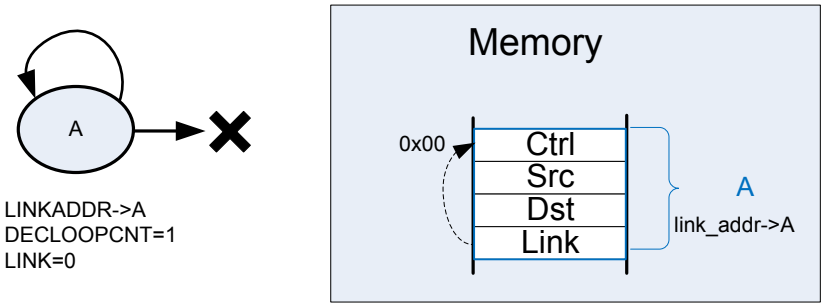


Figure 26.6. Single Descriptor Looped Transfer

Note that the looping descriptor must be stored in memory, because it must load itself from the link address in memory on each iteration.

26.4.4 Descriptor List with Looping

This example uses a descriptor list in memory with looping over multiple descriptors. This example also uses the looping feature and continues on with the next sequential descriptor after looping completes.

The descriptor list in memory is shown in figure Figure 26.7 Descriptor List with Looping on page 942. Descriptor A links to descriptor B. Descriptor B has the DECLOOPCNT bit enabled and loops back to the start of descriptor A. The LINK address of descriptor B is used for the loop address. The LINK bit is set to indicate that execution will continue after completion of looping. Once the LOOPCNT reaches zero, the LDMA will load descriptor C. Descriptor C must be located immediately following descriptor B.

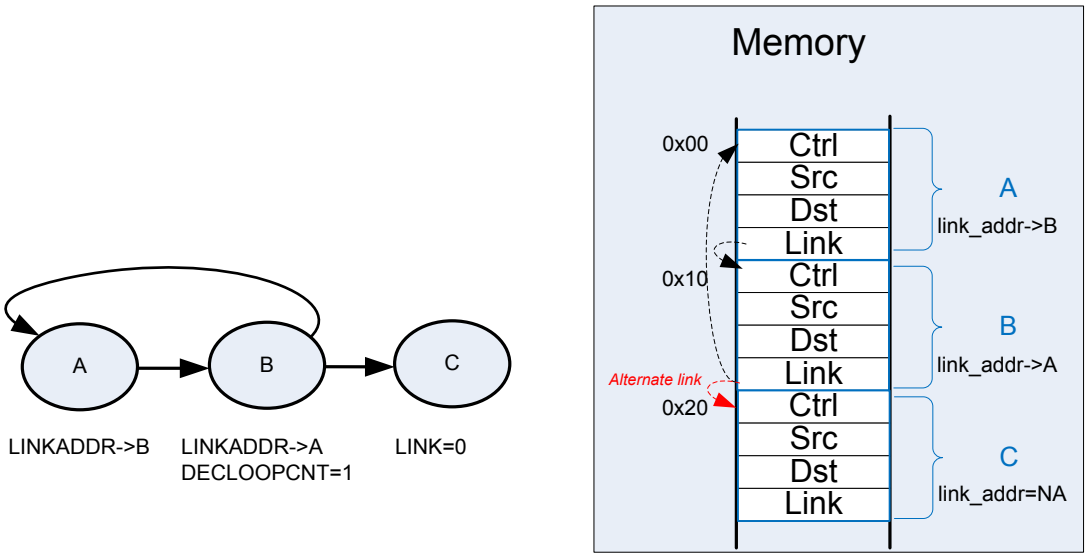


Figure 26.7. Descriptor List with Looping

Initialization is similar to the single looping descriptor with the following modifications.

- Set the LINK bit in descriptors A and B
- write the address of descriptor A into the LIKADDRESS of descriptor B
- write the address of descriptor B into the LIKADDRESS of descriptor A
- Descriptor C must be located immediately after descriptor B in memory

26.4.5 Simple Inter-Channel Synchronization

The LDMA controller features synchronization structures which allow differing channels and/or hardware events to pause a DMA sequence, and wait for a synchronizing event to restart it.

In this example DMA channel 0 and 1 are tasked with the transfer of different sets of data. Channel 0 has two transfer structures, and channel 1 just one, but channel 0 must wait until channel 1 has completed its transfer before it starts its second transfer structure.

Pausing channel 0 is accomplished by inserting a sync wait structure between the two transfer structures. This sync structure waits on SYNCTRIG[7] to be set by a sync set/clear structure which is controlled by channel 1. Sync structures do not transfer data, they can only set, clear, or wait to match the SYNCTRIG[7:0] bits. Note that sync structures cannot decrement loop counter.

```
LDMA_SYNC
    SYNCTRIG=0x0 (at time 0)

LDMA_CH0

    Structure A @ 0x00              Structure B @ 0x10                  Structure C @ 0x20
    CTRL                                CTRL                                CTRL
        STRUCTTYPE=XFER                     STRUCTTYPE=SYNC                     STRUCTTYPE=XFER
    LINK                                LINK                                LINK
        LINKADDR[29:0]=0x00000004           LINKADDR[29:0]=0x00000008           LINKADDR[29:0]=NA
        LINK=1                              LINK=1                              LINK=0
                                        DST
                                            MATCHEN=0x80
                                            MATCHVAL=0x80 (waits for SYNCTRIG[7]=1)

LDMA_CH1

    Structure Y @ 0x30              Structure Z @ 0x40

    CTRL                                CTRL
        STRUCTTYPE=XFER                     STRUCTTYPE=SYNC
    LINK                                LINK
        LINKADDR[29:0]=0x00000010           LINKADDR=NA
        LINK=1                              LINK=0
                                        SRC
                                            SRCCLR=0x0
                                            SRCSET=0x80 (sets SYNCTRIG[7])
```

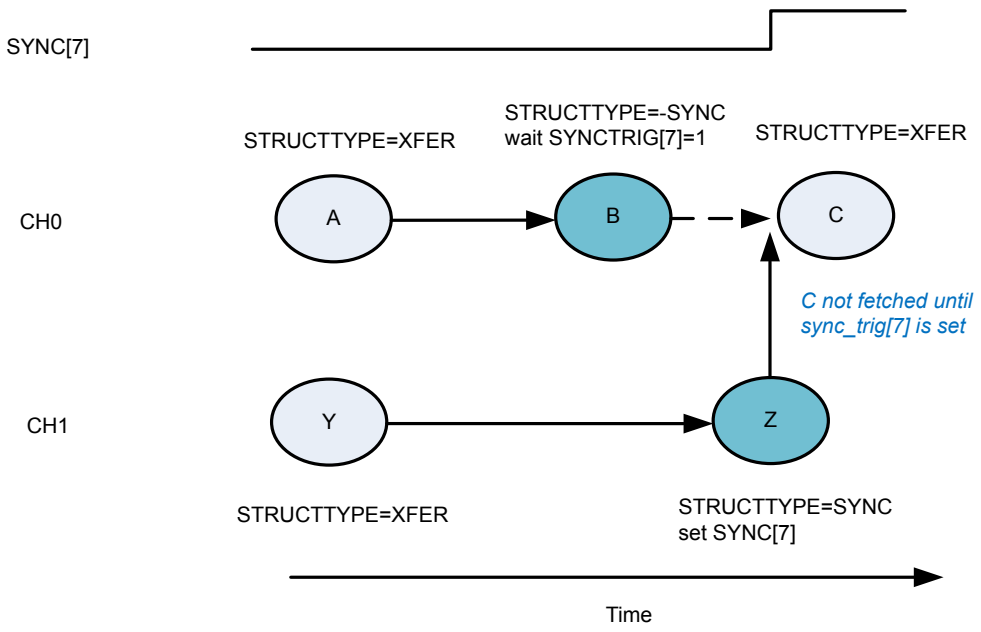


Figure 26.8. Simple Intra-channel Synchronization Example

Both A and Y effectively start at the same time. A finishes earlier, then it links to B, which waits for the SYNCTRIG[7] bit to be set before loading C. Y finishes after B is loaded, and it links to sync structure Z, which sets the SYNCTRIG[7] bit. Channel 0 responds to the trigger set by loading C for the final data transfer.

26.4.6 2D Copy

The LDMA can easily perform a 2D copy using a descriptor list with looping. This set up is visualized in Figure 26.9 2D copy on page 945.

For an application working with graphics, this would mean the ability to copy a rectangle of a given width and height from one picture to another.

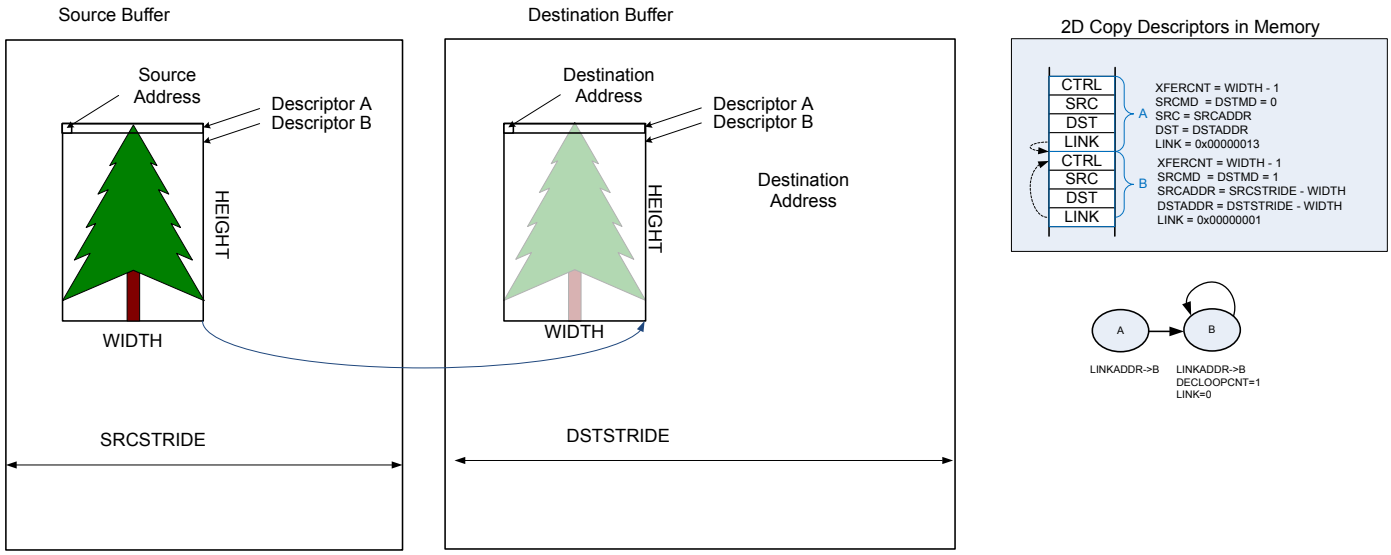


Figure 26.9. 2D copy

The first descriptor will use absolute addressing mode and the source and destination addresses should point to the desired target addresses. The first descriptor will copy only the first row. The XFERCNT of the first descriptor is set to the desired width minus one.

- CTRL
  - XFERCNT = WIDTH - 1
  - SRCMD = 0 (absolute)
  - DSTMD = 0 (absolute)
- SRCADDR = target source address
- DSTADDR = target destination address
- LINK = 0x00000013
  - LINK=1
  - LINKMD=1
  - LINKADDR=0x00000010 (point to next descriptor)

The second descriptor will use relative addressing and the source and destination addresses are set to the desired offset. After the completion of the first descriptor, the address registers will point to the last address transferred. Thus, the width must be subtracted from the stride to get the offset. The second descriptor uses looping and the link register has not offset.

- CTRL
  - XFERCNT = WIDTH - 1
  - SRCMD = 1 (relative)
  - DSTMD = 1 (relative)
  - DECLOOPCNT = 1
- SRCADDR = desired source offset (SRCSTRIDE-WIDTH)
- DSTADDR = desired destination offset (DSTSTRIDE-WIDTH)
- LINK = 0x00000001
  - LINK=0
  - LINKMD=1 (relative)
  - LINKADDR=0x000000000 (no offset)

Because the first descriptor already transferred one row, the number of looping repeats should be the desired height minus two. Therefore, LOOPCNT should be set to HEIGHT minus two before initiating the transfer.

This same method is easily extended to copy multiple rectangles by linking descriptors together. To initialize the LDMA_CHx_LOOP register, precede each descriptor pair described above with a write immediate descriptor which writes the desired value to the LOOPCNT field of the LDMA_CHx_LOOP register.

26.4.7 Ping-Pong

Communication peripherals often use ping-pong buffers. Ping-pong buffers allow the CPU to process data in one buffer while a peripheral transmits or receives data in the other buffer.

Both transmit and receive ping-pong buffers are easily implemented using the LDMA. In either case, this requires two descriptors as shown in Figure 26.10 Infinite Ping-Pong Example on page 947. The LINKADDR field of the LINK member should point to the other descriptor. Using two adjacent descriptors and relative link addressing ensures the descriptors are easily reloadable.

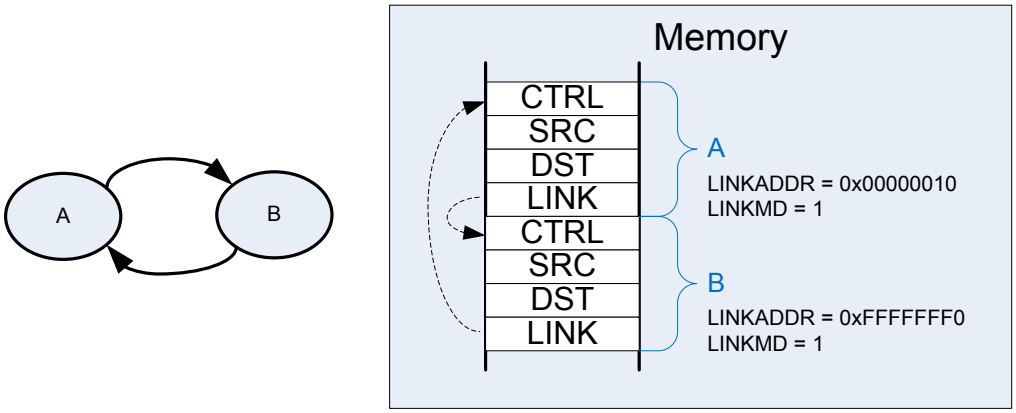


Figure 26.10. Infinite Ping-Pong Example

A **receiver** ping-pong buffer controller consists of two buffers and two descriptors stored in memory that point to the two buffers. Once initialized, as the peripheral receives data, it will fill the first buffer. Once the first buffer is full, it will link automatically to the second buffer and generate an interrupt. Software will then process the data in the first buffer while the LDMA is transferring data to the second buffer. For a receiver ping-pong buffer each descriptor should link to the other descriptor. The link bit should be set to provide infinite ping pong between the two buffers. The DONIFS bit in each descriptor should be set to generate an interrupt on the completion of each descriptor.

- Descriptor A
  - CTRL
    - DONEIFS = 1
    - other settings as desired
  - SRCADDR = peripheral source address
  - DSTADDR = memory destination address
  - LINK = 0x00000013
    - LINKADDR = 0x00000010 (next descriptor)
    - LINK = 1 (link to next descriptor)
    - LINKMD = 1 (relative addressing)
- Descriptor B
  - CTRL
    - DONEIFS = 1
    - other settings as desired
  - SRCADDR = peripheral source address
  - DSTADDR = memory destination address
  - LINK = 0xFFFFFFF3
    - LINKADDR = 0xFFFFFFF0 (previous descriptor)
    - LINK = 1 (link to previous descriptor)
    - LINKMD = 1 (relative addressing)

For **transmitter** ping-pong buffer, software will fill the first buffer and then initiate the DMA transfer. The LDMA will transmit the first buffer data while software is filling the second buffer. In this case, the two descriptors should point to each other, but not automatically

continue to the second buffer. The LINK bit should be cleared to zero. Once software has loaded the first buffer, it will use the LINK-LOAD bit to load the first descriptor and transmit the data. The DONIFS need not be set in each descriptor. The DMA will stop and then generate an interrupt at the completion of each descriptor.

- Descriptor A
  - CTRL
    - DONEIFS = 0
    - other settings as desired
  - SRCADDR = memory source address
  - DSTADDR = peripheral destination address
  - LINK = 0x00000013
    - LINKADDR = 0x00000010 (next descriptor)
    - LINK = 0 (link to next descriptor)
    - LINKMD = 1 (relative addressing)
- Descriptor B
  - CTRL
    - DONEIFS = 0
    - other settings as desired
  - SRCADDR = memory source address
  - DSTADDR = peripheral destination address
  - LINK = 0xFFFFFFF3
    - LINKADDR = 0xFFFFFFF0 (previous descriptor)
    - LINK = 0 (link to previous descriptor)
    - LINKMD = 1 (relative addressing)

26.4.8 Scatter-Gather

Scatter-Gather in general refers to a process that copies data from multiple locations scattered in memory and gathers the data to a single location in memory, or vice versa. A simple descriptor list allows data gathering. For example, data from a discontiguous list of buffers might be copied to a contiguous sequential array of buffers. The inverse is also possible when a sequential array of buffers is scattered to a discontiguous list of available buffers. See section 26.4.2 Descriptor Linked List.

Some DMAs which only have two descriptors implement scatter-gather by using one descriptor to modify the other descriptor. While it is possible to implement this same behavior using the LDMA, it is much more straight-forward to just use a simple descriptor list.

26.5 LDMA Source Selection Details

26.5.1 LDMA Source Selection Details

Table 26.3. LDMA Source Selection Details

SOURCESEL	Source Name	SIGSEL	Request Signal Name
0x0	LDMAXBAR	0x0	LDMAXBAR_DMA_PRSREQ0
		0x1	LDMAXBAR_DMA_PRSREQ1
0x1	TIMER0	0x0	TIMER0_DMA_CC0
		0x1	TIMER0_DMA_CC1
		0x2	TIMER0_DMA_CC2
		0x3	TIMER0_DMA_UFOF
0x2	TIMER1	0x0	TIMER1_DMA_CC0
		0x1	TIMER1_DMA_CC1
		0x2	TIMER1_DMA_CC2
		0x3	TIMER1_DMA_UFOF
0x3	USART0	0x0	USART0_DMA_RXDATAV
		0x1	USART0_DMA_RXDATAVRIGHT
		0x2	USART0_DMA_TXBL
		0x3	USART0_DMA_TXBLRIGHT
		0x4	USART0_DMA_TXEMPTY
0x4	USART1	0x0	USART1_DMA_RXDATAV
		0x1	USART1_DMA_RXDATAVRIGHT
		0x2	USART1_DMA_TXBL
		0x3	USART1_DMA_TXBLRIGHT
		0x4	USART1_DMA_TXEMPTY
0x5	I2C0	0x0	I2C0_DMA_RXDATAV
		0x1	I2C0_DMA_TXBL
0x6	I2C1	0x0	I2C1_DMA_RXDATAV
		0x1	I2C1_DMA_TXBL
0xA	IADC0	0x0	IADC0_DMA_IADC_SCAN
		0x1	IADC0_DMA_IADC_SINGLE
0xB	MSC	0x0	MSC_DMA_WDATA
0xC	TIMER2	0x0	TIMER2_DMA_CC0
		0x1	TIMER2_DMA_CC1
		0x2	TIMER2_DMA_CC2
		0x3	TIMER2_DMA_UFOF

SOURCESEL	Source Name	SIGSEL	Request Signal Name
0xD	TIMER3	0x0	TIMER3_DMA_CC0
		0x1	TIMER3_DMA_CC1
		0x2	TIMER3_DMA_CC2
		0x3	TIMER3_DMA_UFOF
0xE	PDM	0x0	PDM_DMA_RXDATAV
0xF	EUART0	0x0	EUART0_DMA_RXFL
		0x1	EUART0_DMA_TXFL
0x10	TIMER4	0x0	TIMER4_DMA_CC0
		0x1	TIMER4_DMA_CC1
		0x2	TIMER4_DMA_CC2
		0x3	TIMER4_DMA_UFOF

26.6 LDMA Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	LDMA_IPVERSION	R	DMA Channel Request Clear Register
0x004	LDMA_EN	RW	DMA module enable disable Register
0x008	LDMA_CTRL	RW	DMA Control Register
0x00C	LDMA_STATUS	RH	DMA Status Register
0x010	LDMA_SYNCSWSET	W	DMA Sync Trig Sw Set Register
0x014	LDMA_SYNCSWCLR	W	DMA Sync Trig Sw Clear register
0x018	LDMA_SYNCHWEN	RW	DMA Sync HW trigger enable register
0x01C	LDMA_SYNCHWSEL	RW	DMA Sync HW trigger selection register
0x020	LDMA_SYNCSTATUS	RH	DMA Sync Trigger Status Register
0x024	LDMA_CHEN	W	DMA Channel Enable Register
0x028	LDMA_CHDIS	W	DMA Channel Disable Register
0x02C	LDMA_CHSTATUS	RH	DMA Channel Status Register
0x030	LDMA_CHBUSY	RH	DMA Channel Busy Register
0x034	LDMA_CHDONE	RWH INTFLAG	DMA Channel Linking Done Register
0x038	LDMA_DBGHALT	RW	DMA Channel Debug Halt Register
0x03C	LDMA_SWREQ	W	DMA Channel Software Transfer Request
0x040	LDMA_REQDIS	RW	DMA Channel Request Disable Register
0x044	LDMA_REQPEND	RH	DMA Channel Requests Pending Register
0x048	LDMA_LINKLOAD	W	DMA Channel Link Load Register
0x04C	LDMA_REQCLEAR	W	DMA Channel Request Clear Register
0x050	LDMA_IF	RWH INTFLAG	Interrupt Flag Register
0x054	LDMA_IEN	RW	Interrupt Enable Register
0x05C	LDMA_CHx_CFG	RW	Channel Configuration Register
0x060	LDMA_CHx_LOOP	RWH	Channel Loop Counter Register
0x064	LDMA_CHx_CTRL	RWH	Channel Descriptor Control Word Register
0x068	LDMA_CHx_SRC	RWH	Channel Descriptor Source Address
0x06C	LDMA_CHx_DST	RWH	Channel Descriptor Destination Address
0x070	LDMA_CHx_LINK	RWH	Channel Descriptor Link Address
0x1000	LDMA_IPVERSION_SET	R	DMA Channel Request Clear Register
0x1004	LDMA_EN_SET	RW	DMA module enable disable Register
0x1008	LDMA_CTRL_SET	RW	DMA Control Register
0x100C	LDMA_STATUS_SET	RH	DMA Status Register
0x1010	LDMA_SYNCSWSET_SET	W	DMA Sync Trig Sw Set Register
0x1014	LDMA_SYNCSWCLR_SET	W	DMA Sync Trig Sw Clear register
0x1018	LDMA_SYNCHWEN_SET	RW	DMA Sync HW trigger enable register

Offset	Name	Type	Description
0x101C	LDMA_SYNCHWSEL_SET	RW	DMA Sync HW trigger selection register
0x1020	LDMA_SYNCSTATUS_SET	RH	DMA Sync Trigger Status Register
0x1024	LDMA_CHEN_SET	W	DMA Channel Enable Register
0x1028	LDMA_CHDIS_SET	W	DMA Channel Disable Register
0x102C	LDMA_CHSTATUS_SET	RH	DMA Channel Status Register
0x1030	LDMA_CHBUSY_SET	RH	DMA Channel Busy Register
0x1034	LDMA_CHDONE_SET	RWH INTFLAG	DMA Channel Linking Done Register
0x1038	LDMA_DBGHALT_SET	RW	DMA Channel Debug Halt Register
0x103C	LDMA_SWREQ_SET	W	DMA Channel Software Transfer Request
0x1040	LDMA_REQDIS_SET	RW	DMA Channel Request Disable Register
0x1044	LDMA_REQPEND_SET	RH	DMA Channel Requests Pending Register
0x1048	LDMA_LINKLOAD_SET	W	DMA Channel Link Load Register
0x104C	LDMA_REQCLEAR_SET	W	DMA Channel Request Clear Register
0x1050	LDMA_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x1054	LDMA_IEN_SET	RW	Interrupt Enable Register
0x105C	LDMA_CHx_CFG_SET	RW	Channel Configuration Register
0x1060	LDMA_CHx_LOOP_SET	RWH	Channel Loop Counter Register
0x1064	LDMA_CHx_CTRL_SET	RWH	Channel Descriptor Control Word Register
0x1068	LDMA_CHx_SRC_SET	RWH	Channel Descriptor Source Address
0x106C	LDMA_CHx_DST_SET	RWH	Channel Descriptor Destination Address
0x1070	LDMA_CHx_LINK_SET	RWH	Channel Descriptor Link Address
0x2000	LDMA_IPVERSION_CLR	R	DMA Channel Request Clear Register
0x2004	LDMA_EN_CLR	RW	DMA module enable disable Register
0x2008	LDMA_CTRL_CLR	RW	DMA Control Register
0x200C	LDMA_STATUS_CLR	RH	DMA Status Register
0x2010	LDMA_SYNCSWSET_CLR	W	DMA Sync Trig Sw Set Register
0x2014	LDMA_SYNCSWCLR_CLR	W	DMA Sync Trig Sw Clear register
0x2018	LDMA_SYNCHWEN_CLR	RW	DMA Sync HW trigger enable register
0x201C	LDMA_SYNCHWSEL_CLR	RW	DMA Sync HW trigger selection register
0x2020	LDMA_SYNCSTATUS_CLR	RH	DMA Sync Trigger Status Register
0x2024	LDMA_CHEN_CLR	W	DMA Channel Enable Register
0x2028	LDMA_CHDIS_CLR	W	DMA Channel Disable Register
0x202C	LDMA_CHSTATUS_CLR	RH	DMA Channel Status Register
0x2030	LDMA_CHBUSY_CLR	RH	DMA Channel Busy Register
0x2034	LDMA_CHDONE_CLR	RWH INTFLAG	DMA Channel Linking Done Register
0x2038	LDMA_DBGHALT_CLR	RW	DMA Channel Debug Halt Register
0x203C	LDMA_SWREQ_CLR	W	DMA Channel Software Transfer Request

Offset	Name	Type	Description
0x2040	LDMA_REQDIS_CLR	RW	DMA Channel Request Disable Register
0x2044	LDMA_REQPEND_CLR	RH	DMA Channel Requests Pending Register
0x2048	LDMA_LINKLOAD_CLR	W	DMA Channel Link Load Register
0x204C	LDMA_REQCLEAR_CLR	W	DMA Channel Request Clear Register
0x2050	LDMA_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x2054	LDMA_IEN_CLR	RW	Interrupt Enable Register
0x205C	LDMA_CHx_CFG_CLR	RW	Channel Configuration Register
0x2060	LDMA_CHx_LOOP_CLR	RWH	Channel Loop Counter Register
0x2064	LDMA_CHx_CTRL_CLR	RWH	Channel Descriptor Control Word Register
0x2068	LDMA_CHx_SRC_CLR	RWH	Channel Descriptor Source Address
0x206C	LDMA_CHx_DST_CLR	RWH	Channel Descriptor Destination Address
0x2070	LDMA_CHx_LINK_CLR	RWH	Channel Descriptor Link Address
0x3000	LDMA_IPVERSION_TGL	R	DMA Channel Request Clear Register
0x3004	LDMA_EN_TGL	RW	DMA module enable disable Register
0x3008	LDMA_CTRL_TGL	RW	DMA Control Register
0x300C	LDMA_STATUS_TGL	RH	DMA Status Register
0x3010	LDMA_SYNCSWSET_TGL	W	DMA Sync Trig Sw Set Register
0x3014	LDMA_SYNCSWCLR_TGL	W	DMA Sync Trig Sw Clear register
0x3018	LDMA_SYNCHWEN_TGL	RW	DMA Sync HW trigger enable register
0x301C	LDMA_SYNCHWSEL_TGL	RW	DMA Sync HW trigger selection register
0x3020	LDMA_SYNCSTATUS_TGL	RH	DMA Sync Trigger Status Register
0x3024	LDMA_CHEN_TGL	W	DMA Channel Enable Register
0x3028	LDMA_CHDIS_TGL	W	DMA Channel Disable Register
0x302C	LDMA_CHSTATUS_TGL	RH	DMA Channel Status Register
0x3030	LDMA_CHBUSY_TGL	RH	DMA Channel Busy Register
0x3034	LDMA_CHDONE_TGL	RWH INTFLAG	DMA Channel Linking Done Register
0x3038	LDMA_DBGHALT_TGL	RW	DMA Channel Debug Halt Register
0x303C	LDMA_SWREQ_TGL	W	DMA Channel Software Transfer Request
0x3040	LDMA_REQDIS_TGL	RW	DMA Channel Request Disable Register
0x3044	LDMA_REQPEND_TGL	RH	DMA Channel Requests Pending Register
0x3048	LDMA_LINKLOAD_TGL	W	DMA Channel Link Load Register
0x304C	LDMA_REQCLEAR_TGL	W	DMA Channel Request Clear Register
0x3050	LDMA_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x3054	LDMA_IEN_TGL	RW	Interrupt Enable Register
0x305C	LDMA_CHx_CFG_TGL	RW	Channel Configuration Register
0x3060	LDMA_CHx_LOOP_TGL	RWH	Channel Loop Counter Register
0x3064	LDMA_CHx_CTRL_TGL	RWH	Channel Descriptor Control Word Register

Offset	Name	Type	Description
0x3068	LDMA_CHx_SRC_TGL	RWH	Channel Descriptor Source Address
0x306C	LDMA_CHx_DST_TGL	RWH	Channel Descriptor Destination Address
0x3070	LDMA_CHx_LINK_TGL	RWH	Channel Descriptor Link Address

26.7 LDMA Register Description

26.7.1 LDMA_IPVERSION - DMA Channel Request Clear Register

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												R				
Name																												IPVERSION				

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	IPVERSION	0x0	R	**IPVERSION**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

26.7.2 LDMA_EN - DMA module enable disable Register

Offset	Bit Position																													
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**LDMA module enable and disable register**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

26.7.3 LDMA_CTRL - DMA Control Register

Offset	Bit Position																															
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0			0x1E																												
Access	RW			RW																												
Name	CORERST			NUMFIXED																												

Bit	Name	Reset	Access	Description
31	CORERST	0x0	RW	**Reset DMA controller**
	Trigger a reset of the LDMA controller core without losing register configuration			
30:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28:24	NUMFIXED	0x1E	RW	**Number of Fixed Priority Channels**
	This field defines the number of Fixed Priority Arbitration channels. Channels CH0 though CH(n-1) are fixed, and channels CH(n) through CH7 are round robin, where n is the field value. The reset value will give all fixed channels.			
23:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

26.7.4 LDMA_STATUS - DMA Status Register

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x1F										0x10						0x0					0x0						0x0	0x0
Access				R										R						R					R						R	R
Name				CHNUM										FIFOLEVEL						CHERROR					CHGRANT						ANYREQ	ANYBUSY

Bit	Name	Reset	Access	Description
31:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28:24	CHNUM	0x1F	R	**Number of Channels**
	The value of CHNUM always reads the total number of channels present for this instance of the DMA controller module.			
23:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20:16	FIFOLEVEL	0x10	R	**FIFO Level**
	The value of FIFOLEVEL indicates the number of entries currently in the FIFO. (Note when all channels are disabled, this register will read the total number of entries in the FIFO.)			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:8	CHERROR	0x0	R	**Errant Channel Number**
	When the ERROR flag is set in the LDMA_IF register, the CHERROR field will indicate the most recent channel to have a transfer error.			
7:3	CHGRANT	0x0	R	**Granted Channel Number**
	The value of this field indicates the currently active channel or last active channel. Note that the reset value for this field is zero.			
2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	ANYREQ	0x0	R	**Any DMA Channel Request Pending**
	The value of this bit will be TRUE (1) if any requests are pending			
0	ANYBUSY	0x0	R	**Any DMA Channel Busy**
	The value of this bit will be TRUE (1) if one or more DMA channels are actively transferring data			

26.7.5 LDMA_SYNCSWSET - DMA Sync Trig Sw Set Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									SYNCSWSET							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	SYNCSWSET	0x0	W	**DMA SYNC Software Trigger Set**
	Sets the corresponding bit in the SYNCSTATUS.SYNCTRIG field to value 1.			

26.7.6 LDMA_SYNCSWCLR - DMA Sync Trig Sw Clear register

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									SYNCSWCLR							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	SYNCSWCLR	0x0	W	**DMA SYNC Software Trigger Clear**
	Clears the corresponding bit in the SYNCSTATUS.SYNCTRIG field to value 0.			

26.7.7 LDMA_SYNCHWEN - DMA Sync HW trigger enable register

Offset	Bit Position																													
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0																0x0							
Access									RW																RW							
Name									SYNCCLREN																SYNCSETEN							

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:16	SYNCCLREN	0x0	RW	**Hardware Sync Trigger Clear Enable**
	Enables the corresponding bit in the SYNCSTATUS.SYNCTRIG field to be cleared by PRS channel 7-0, mapping to bits [23:16].			
15:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	SYNCSETEN	0x0	RW	**Hardware Sync Trigger Set Enable**
	Enables the corresponding bit in the SYNCSTATUS.SYNCTRIG field to be set by PRS channel 7-0, mapping to bits [7:0].			

26.7.8 LDMA_SYNCHWSEL - DMA Sync HW trigger selection register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x0																0x0				
Access												RW																RW				
Name												SYNCCLREDGE																SYNCSETEDGE				

Bit	Name	Reset	Access	Description
31:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:16	SYNCCLREDGE	0x0	RW	**Hardware Sync Trigger Clear Edge Select**

Select rising or falling edge detection on PRS to clear trigger.

Value	Mode	Description
0	RISE	Use rising edge detection
1	FALL	Use falling edge detection

Bit	Name	Reset	Access	Description
15:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	SYNCSETEDGE	0x0	RW	**Hardware Sync Trigger Set Edge Select**

Select rising or falling edge detection on PRS to set trigger.

Value	Mode	Description
0	RISE	Use rising edge detection
1	FALL	Use falling edge detection

26.7.9 LDMA_SYNCSTATUS - DMA Sync Trigger Status Register

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									SYNCTRIG							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	SYNCTRIG	0x0	R	**sync trig status**
	Reflects the status of setting and clearing by software (SYNCSWSET/SYNCSWCLR), hardware (PRS), and loading SYNC structures. Setting a bit always takes prescedence over clearing.			

26.7.10 LDMA_CHEN - DMA Channel Enable Register

Offset	Bit Position																															
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									CHEN							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	CHEN	0x0	W	**Channel Enables**
	Setting one of these bits will enable the respective DMA channel, writing zeros has no effect			

26.7.11 LDMA_CHDIS - DMA Channel Disable Register

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									CHDIS							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	CHDIS	0x0	W	**DMA Channel disable**
	Setting one of these bits will disable respective DMA channel, writing zeros has no effect. If set while a transfer is in progress, the current transfer block will complete. The remaining blocks will pause until resumed later by setting corresponding CHEN bit.			

26.7.12 LDMA_CHSTATUS - DMA Channel Status Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									CHSTATUS							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	CHSTATUS	0x0	R	**DMA Channel Status**
	The value of this bit will be TRUE (1) if one or more DMA channels are enabled			

26.7.13 LDMA_CHBUSY - DMA Channel Busy Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												R				
Name																												BUSY				

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	BUSY	0x0	R	**Channels Busy** The bits of this field read 1 when the corresponding channel is busy.

26.7.14 LDMA_CHDONE - DMA Channel Linking Done Register

Offset	Bit Position																													
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access																									RW	RW	RW	RW	RW	RW	RW	RW
Name																									CHDONE7	CHDONE6	CHDONE5	CHDONE4	CHDONE3	CHDONE2	CHDONE1	CHDONE0

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	CHDONE7	0x0	RW	**DMA Channel Link done intr flag**
	Each DMA channel sets the corresponding bit in this register when the entire linked transfer is done. The interrupt service routine should clear these bits. For non-link structures this will be set once the structure is done.			
6	CHDONE6	0x0	RW	**DMA Channel Link done intr flag**
	Each DMA channel sets the corresponding bit in this register when the entire linked transfer is done. The interrupt service routine should clear these bits. For non-link structures this will be set once the structure is done.			
5	CHDONE5	0x0	RW	**DMA Channel Link done intr flag**
	Each DMA channel sets the corresponding bit in this register when the entire linked transfer is done. The interrupt service routine should clear these bits. For non-link structures this will be set once the structure is done.			
4	CHDONE4	0x0	RW	**DMA Channel Link done intr flag**
	Each DMA channel sets the corresponding bit in this register when the entire linked transfer is done. The interrupt service routine should clear these bits. For non-link structures this will be set once the structure is done.			
3	CHDONE3	0x0	RW	**DMA Channel Link done intr flag**
	Each DMA channel sets the corresponding bit in this register when the entire linked transfer is done. The interrupt service routine should clear these bits. For non-link structures this will be set once the structure is done.			
2	CHDONE2	0x0	RW	**DMA Channel Link done intr flag**
	Each DMA channel sets the corresponding bit in this register when the entire linked transfer is done. The interrupt service routine should clear these bits. For non-link structures this will be set once the structure is done.			
1	CHDONE1	0x0	RW	**DMA Channel Link done intr flag**
	Each DMA channel sets the corresponding bit in this register when the entire linked transfer is done. The interrupt service routine should clear these bits. For non-link structures this will be set once the structure is done.			
0	CHDONE0	0x0	RW	**DMA Channel Link done intr flag**
	Each DMA channel sets the corresponding bit in this register when the entire linked transfer is done. The interrupt service routine should clear these bits. For non-link structures this will be set once the structure is done.			

26.7.15 LDMA_DBGHALT - DMA Channel Debug Halt Register

Offset	Bit Position																													
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												RW				
Name																												DBGHALT				

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	DBGHALT	0x0	RW	**DMA Debug Halt**
	Setting one of these bits will mask the corresponding DMA channel's peripheral request when debugging and the CPU is halted. This may be useful for debugging DMA software.			

26.7.16 LDMA_SWREQ - DMA Channel Software Transfer Request

Offset	Bit Position																													
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0				
Access																												W				
Name																												SWREQ				

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	SWREQ	0x0	W	**Software Transfer Requests**
	Setting one of these bits will trigger a DMA transfer for the corresponding channel. Writing zeros has no effect.			

26.7.17 LDMA_REQDIS - DMA Channel Request Disable Register

Offset	Bit Position																													
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									RW							
Name																									REQDIS							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	REQDIS	0x0	RW	**DMA Request Disables**
	Setting one of these bits will disable peripheral requests for the corresponding channel. When cleared any pending peripheral requests will be serviced.			

26.7.18 LDMA_REQPEND - DMA Channel Requests Pending Register

Offset	Bit Position																													
0x044	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									R							
Name																									REQPEND							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	REQPEND	0x0	R	**DMA Requests Pending**
	When a DMA channel has a pending peripheral request the corresponding REQPEND bit will read 1.			

26.7.19 LDMA_LINKLOAD - DMA Channel Link Load Register

Offset	Bit Position																															
0x048	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									LINKLOAD							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	LINKLOAD	0x0	W	**DMA Link Loads** Setting one of these bits will force the corresponding DMA channel to load the next DMA structure and enable the channel. This empowers software to step through a sequence of descriptors.

26.7.20 LDMA_REQCLEAR - DMA Channel Request Clear Register

Offset	Bit Position																															
0x04C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									W							
Name																									REQCLEAR							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	REQCLEAR	0x0	W	**DMA Request Clear** Setting one of these bits will clear any internally registered transfer requests for the corresponding channel.

26.7.21 LDMA_IF - Interrupt Flag Register

Offset	Bit Position																													
0x050	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																								0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access	RW																								RW	RW	RW	RW	RW	RW	RW	RW
Name	ERROR																								DONE7	DONE6	DONE5	DONE4	DONE3	DONE2	DONE1	DONE0

Bit	Name	Reset	Access	Description
31	ERROR	0x0	RW	**Error Flag**
	Set to 1 on an Error			
30:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	DONE7	0x0	RW	**DMA Structure Operation Done**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set the LDMA_IF register.			
6	DONE6	0x0	RW	**DMA Structure Operation Done**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set the LDMA_IF register.			
5	DONE5	0x0	RW	**DMA Structure Operation Done**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set the LDMA_IF register.			
4	DONE4	0x0	RW	**DMA Structure Operation Done**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set the LDMA_IF register.			
3	DONE3	0x0	RW	**DMA Structure Operation Done**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set the LDMA_IF register.			
2	DONE2	0x0	RW	**DMA Structure Operation Done**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set the LDMA_IF register.			
1	DONE1	0x0	RW	**DMA Structure Operation Done**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set the LDMA_IF register.			
0	DONE0	0x0	RW	**DMA Structure Operation Done**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set the LDMA_IF register.			

26.7.22 LDMA_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x054	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																											0x0				
Access	RW																											RW				
Name	ERROR																											CHDONE				

Bit	Name	Reset	Access	Description
31	ERROR	0x0	RW	**Enable or disable the error interrupt**
	This is the bitfield to enable the link done interrupt			
30:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	CHDONE	0x0	RW	**Enable or disable the done interrupt**
	This is the bitfield to enable the AHB bus error interrupt			

26.7.23 LDMA_CHx_CFG - Channel Configuration Register

Offset	Bit Position																													
0x05C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x0	0x0			0x0																
Access											RW	RW			RW																
Name											DSTINCSIGN	SRCINCSIGN			ARBSLOTS																

Bit	Name	Reset	Access	Description
31:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	DSTINCSIGN	0x0	RW	**Destination Address Increment Sign**
	0: Increment destination address, 1: Decrement destination address			
	Value	Mode		Description
	0	POSITIVE		Increment destination address
	1	NEGATIVE		Decrement destination address
20	SRCINCSIGN	0x0	RW	**Source Address Increment Sign**
	0: Increment source address, 1: Decrement source address			
	Value	Mode		Description
	0	POSITIVE		Increment source address
	1	NEGATIVE		Decrement source address
19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	ARBSLOTS	0x0	RW	**Arbitration Slot Number Select**
	For channels using round robin arbitration, this bit-field is used to select the number of slots in the round robin queue.			
	Value	Mode		Description
	0	ONE		One arbitration slot selected
	1	TWO		Two arbitration slots selected
	2	FOUR		Four arbitration slots selected
	3	EIGHT		Eight arbitration slots selected
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

26.7.24 LDMA_CHx_LOOP - Channel Loop Counter Register

Offset	Bit Position																													
0x060	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0							
Access																									RW							
Name																									LOOPCNT							

Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	LOOPCNT	0x0	RW	**Linked Structure Sequence Loop Counter** This bit-field specifies the number of iterations when using looping descriptors. Software should write to LOOPCNT before using a looping descriptor.

26.7.25 LDMA_CHx_CTRL - Channel Descriptor Control Word Register

Offset	Bit Position																															
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0	0x0	0x0		0x0		0x0		0x0	0x0	0x0	0x0	0x0				0x0	0x0											0x0		0x0	
Access	R	R	RW		RW		RW		RW	RW	RW	RW	RW				RW	RW											R		RW	
Name	DSTMODE	SRCMODE	DSTINC		SIZE		SRCINC		IGNORESREQ	DECLOOPCNT	REQMODE	DONEIEN	BLOCKSIZE				BYTESWAP	XFERCNT											STRUCTREQ		STRUCTTYPE	

Bit	Name	Reset	Access	Description
31	DSTMODE	0x0	R	**Destination Addressing Mode**

This field specifies the destination addressing mode of linked descriptors. After loading a linked descriptor, reading this field will indicate the destination addressing mode of the linked descriptor. Note that the first descriptor always uses absolute addressing mode.

Value	Mode	Description
0	ABSOLUTE	The DSTADDR field of LDMA_CHx_DST contains the absolute address of the destination data.
1	RELATIVE	The DSTADDR field of LDMA_CHx_DST contains the relative offset of the destination data.

Bit	Name	Reset	Access	Description
30	SRCMODE	0x0	R	**Source Addressing Mode**

This field specifies the source addressing mode of linked descriptors. After loading a linked descriptor, reading this field will indicate the source addressing mode of the linked descriptor. Note that the first descriptor always uses absolute addressing mode.

Value	Mode	Description
0	ABSOLUTE	The SRCADDR field of LDMA_CHx_SRC contains the absolute address of the source data.
1	RELATIVE	The SRCADDR field of LDMA_CHx_SRC contains the relative offset of the source data.

Bit	Name	Reset	Access	Description
29:28	DSTINC	0x0	RW	**Destination Address Increment Size**

This bit-field specifies the stride or number of unit data addresses to increment the destination address after each unit of data is transferred. The unit data width is controlled by the SIZE bit-field and can be a byte, half-word or word.

Value	Mode	Description
0	ONE	Increment destination address by one unit data size after each write
1	TWO	Increment destination address by two unit data sizes after each write
2	FOUR	Increment destination address by four unit data sizes after each write
3	NONE	Do not increment the destination address. Writes are made to a fixed destination address, for example writing to a FIFO.

Bit	Name	Reset	Access	Description
27:26	SIZE	0x0	RW	**Unit Data Transfer Size**

This field specifies the size of data transferred.

Value	Mode	Description
0	BYTE	Each unit transfer is a byte
1	HALFWORD	Each unit transfer is a half-word
2	WORD	Each unit transfer is a word

25:24	SRCINC	0x0	RW	**Source Address Increment Size**

This bit-field specifies the stride or number of unit data addresses to increment the source address after each unit of data is transferred. The unit data width is controlled by the SIZE bit-field and can be a byte, half-word or word.

Value	Mode	Description
0	ONE	Increment source address by one unit data size after each read
1	TWO	Increment source address by two unit data sizes after each read
2	FOUR	Increment source address by four unit data sizes after each read
3	NONE	Do not increment the source address. In this mode reads are made from a fixed source address, for example reading FIFO.

23	IGNORESREQ	0x0	RW	**Ignore Sreq**

The channel arbiter will ignore single requests (SREQ) and only respond to multiple requests (REQ) when this bit is set.

22	DECLOOPCNT	0x0	RW	**Decrement Loop Count**

When using looping, setting this bit will decrement the LOOPCNT field in the LDMA_CHx_LOOP register after each descriptor execution.

21	REQMODE	0x0	RW	**DMA Request Transfer Mode Select**

Selects the DMA Request Transfer mode.

Value	Mode	Description
0	BLOCK	The LDMA transfers one BLOCKSIZE per transfer request.
1	ALL	One transfer request transfers all units as defined by the XFRCNT field.

20	DONEIEN	0x0	RW	**DMA Operation Done Interrupt Flag Set En**

Setting this bit will set the interrupt flag when the transfer is done, or linked in the case where the LINK bit is set, or synchronized in the case of a SYNC transfer.

19:16	BLOCKSIZE	0x0	RW	**Block Transfer Size**

This bit-field controls the number of unit data transfers per arbitration cycle

Value	Mode	Description
0	UNIT1	One unit transfer per arbitration
1	UNIT2	Two unit transfers per arbitration
2	UNIT3	Three unit transfers per arbitration
3	UNIT4	Four unit transfers per arbitration

Bit	Name	Reset	Access	Description
	4	UNIT6		Six unit transfers per arbitration
	5	UNIT8		Eight unit transfers per arbitration
	7	UNIT16		Sixteen unit transfers per arbitration
	9	UNIT32		32 unit transfers per arbitration
	10	UNIT64		64 unit transfers per arbitration
	11	UNIT128		128 unit transfers per arbitration
	12	UNIT256		256 unit transfers per arbitration
	13	UNIT512		512 unit transfers per arbitration
	14	UNIT1024		1024 unit transfers per arbitration
	15	ALL		Transfer all units as specified by the XFRCNT field
15	BYTESWAP	0x0	RW	**Endian Byte Swap**
	For word and half-word transfers, setting this bit will swap all bytes of each word or half-word.			
14:4	XFERCNT	0x0	RW	**DMA Unit Data Transfer Count**
	Specifies number of unit data (words, half-words, or bytes) to transfer, as determined by the SIZE field. The value written should be one less than the desired transfer count.			
3	STRUCTREQ	0x0	R	**Structure DMA Transfer Request**
	Structure Transfer Request			
2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	STRUCTTYPE	0x0	RW	**DMA Structure Type**
	DMA Structure type			

Value	Mode	Description
0	TRANSFER	DMA transfer structure type selected.
1	SYNCHRONIZE	Synchronization structure type selected.
2	WRITE	Write immediate value structure type selected.

26.7.26 LDMA_CHx_SRC - Channel Descriptor Source Address

Offset	Bit Position																															
0x068	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	SRCADDR																															

Bit	Name	Reset	Access	Description
31:0	SRCADDR	0x0	RW	**Source Data Address**
	Writing to this register sets the source address. Reading from this register during a DMA transfer will indicate the next source read address. The value of this register is incremented or decremented with each source read.			

26.7.27 LDMA_CHx_DST - Channel Descriptor Destination Address

Offset	Bit Position																															
0x06C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RW																															
Name	DSTADDR																															

Bit	Name	Reset	Access	Description
31:0	DSTADDR	0x0	RW	**Destination Data Address**
	Writing to this register sets the destination address. Reading from this register during a DMA transfer will indicate the next destination write address. This value of this register is incremented or decremented with each destination write.			

26.7.28 LDMA_CHx_LINK - Channel Descriptor Link Address

Offset	Bit Position																													
0x070	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																														0x0	0x0
Access	RW																														RW	R
Name	LINKADDR																														LINK	LINKMODE

Bit	Name	Reset	Access	Description
31:2	LINKADDR	0x0	RW	**Link Structure Address**
	To use linking, write the address of the the first linked descriptor to this register. When a linked descriptor is loaded, it may also be linked to another descriptor. Reading this register will reflect the address of the next linked descriptor.			
1	LINK	0x0	RW	**Link Next Structure**
	After completing the initial transfer, if this bit is NOT set, the DMA will load the next linked descriptor. If the next linked descriptor also has this bit set, the DMA will load the next linked descriptor.			
0	LINKMODE	0x0	R	**Link Structure Addressing Mode**
	This field specifies the addressing mode of linked descriptors. After loading a linked descriptor, reading this field will indicate the addressing mode of the loaded linked descriptor. Note that the first descriptor always uses absolute addressing mode.			

Value	Mode	Description
0	ABSOLUTE	The LINKADDR field of LDMA_CHx_LINK contains the absolute address of the linked descriptor.
1	RELATIVE	The LINKADDR field of LDMA_CHx_LINK contains the relative offset of the linked descriptor.

26.8 LDMAXBAR Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	LDMAXBAR_CHx_REQSEL	RW	Channel Peripheral Request Select Reg...
0x1000	LDMAXBAR_CHx_REQSEL_SET	RW	Channel Peripheral Request Select Reg...
0x2000	LDMAXBAR_CHx_REQSEL_CLR	RW	Channel Peripheral Request Select Reg...
0x3000	LDMAXBAR_CHx_REQSEL_TGL	RW	Channel Peripheral Request Select Reg...

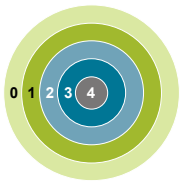
26.9 LDMAXBAR Register Description

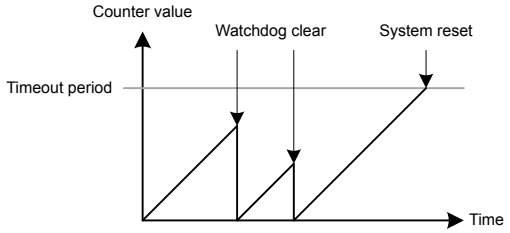
26.9.1 LDMAXBAR_CHx_REQSEL - Channel Peripheral Request Select Reg...

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x0																		0x0			
Access											RW																		RW			
Name											SOURCESEL																		SIGSEL			

Bit	Name	Reset	Access	Description
31:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:16	SOURCESEL	0x0	RW	**Source Select** Select input source to DMA channel.
15:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	SIGSEL	0x0	RW	**Signal Select** Select input signal to DMA channel.

27. WDOG - Watch Dog Timer





Quick Facts

What?

The WDOG (Watchdog Timer) resets the system in case of a fault condition, and can be enabled in all energy modes as long as the low frequency clock source is available.

Why?

If a software failure or external event renders the MCU unresponsive, a Watchdog timeout will reset the system to a known, safe state..

How?

An enabled Watchdog Timer implements a configurable timeout period. If the CPU fails to re-start the Watchdog Timer before it times out, a full system reset will be triggered. The Watchdog consumes insignificant power, and allows the device to remain safely in low energy modes for up to 256 seconds at a time.

27.1 Introduction

The purpose of the watchdog timer is to generate a reset in case of a system failure to increase application reliability. The failure can be caused by a variety of events, such as an ESD pulse or a software failure.

27.2 Features

- Clock input from selectable oscillators
  - Internal 32 kHz LFRCO oscillator
  - Internal 1 kHz ULFRCO oscillator
  - External 32.768 kHz LFXO XTAL oscillator
  - HCLK divided by 1024
- Configurable timeout period from 9 to 256k watchdog clock cycles
- Individual selection to keep running or freeze when entering EM2 DeepSleep or EM3 Stop
- Selection to keep running or freeze when entering debug mode
- Selection to block the CPU from entering Energy Mode 4
- Configurable warning interrupt at 25%,50%, or 75% of the timeout period
- Configurable window interrupt at 12.5%,25%,37.5%,50%,62.5%,75%,87.5% of the timeout period
- Timeout interrupt
- PRS as a watchdog clear
- Interrupt for the event where a PRS rising edge is absent before a software reset

27.3 Functional Description

The watchdog is enabled by setting the EN bit in WDOGn_EN. When enabled, the watchdog counts up to the period value configured through the PERSEL field in WDOGn_CFG. If the watchdog timer is not cleared to 0 (by writing a 1 to the CLEAR bit in WDOGn_CMD) before the period is reached, the chip is reset. If a timely clear command is issued, the timer starts counting up from 0 again. The watchdog can optionally be locked by writing anything other than UNLOCK code in WDOGn_LOCK. Once locked, it cannot be disabled or reconfigured by software.

When the EN bit in WDOGn_EN cleared to 0, the watchdog counter is reset.

27.3.1 Clock Source

Four clock sources are available for use with the watchdog, through the CLKSEL field in CMU_WDOGn_CFG. The selected oscillator source automatically starts when the watchdog is enabled. To prevent accidental change of the clock selection, CMU_WDOGLOCK can be written anything other than UNLOCK code. Also, respective oscillator has locks to prevent accidental disabling of oscillators. The PERSEL field in WDOGn_CFG is used to divide the selected watchdog clock, and the timeout for the watchdog timer can be calculated with the formula:

$$T_{TIMEOUT} = [2^{(PERSEL+3)} + 1] / f$$

where f is the frequency of the selected clock.

Users must clear EM2RUN and EM3RUN when the selected clock source is HFCLKDIV1024.

27.3.2 Debug Functionality

The watchdog timer can either keep running or be frozen when the device is halted by a debugger. This configuration is done through the DEBUGRUN bit in WDOGn_CFG. When code execution is resumed, the watchdog will continue counting where it left off.

27.3.3 Energy Mode Handling

The watchdog timer can be configured to either keep on running or freeze when entering EM2 DeepSleep or EM3 Stop. The configuration is done individually for each energy mode in the EM2RUN and EM3RUN bits in WDOGn_CFG. When the watchdog has been frozen and is re-entering an energy mode where it is running, the watchdog timer will continue counting where it left off. For the watchdog there is no difference between EM0 Active and EM1 Sleep. The watchdog does not run in EM4. If EM4BLOCK in WDOGn_CFG is set, the CPU will be prevented from entering EM4 by software request.

27.3.4 Warning Interrupt

The watchdog implements a warning interrupt which can be configured to occur at approximately 25%, 50%, or 75% of the timeout period through the WARNSEL field of the WDOGn_CFG register. This interrupt can be used to wake up the cpu for clearing the watchdog. The warning point for the watchdog timer can be calculated with the formula:

$$T_{WARNING} = [2^{(PERSEL+3)} +1]*WARNSEL / 4 / f$$

where f is the frequency of the selected clock.

When the watchdog is enabled, it is recommended to clear the watchdog before changing WARNSEL.

27.3.5 Window Interrupt

This interrupt occurs when the watchdog is cleared below a certain threshold. This threshold is given by the formula:

$$T_{WINDOW} = [2^{(PERSEL+3)} + 1]*WARNSEL / 8 / f$$

where f is the frequency of the selected clock.

This value will be approximately 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, or 87.5% of the timeout value based on the WINSEL field of the WDOGn_CFG. Figure 27.2 WDOG Warning, Window, and Timeout on page 979 illustrates the warning, the window, and the timeout interrupts. Also, it shows where the prs rising edge needs to happen. The prs edge detection feature is discussed later.

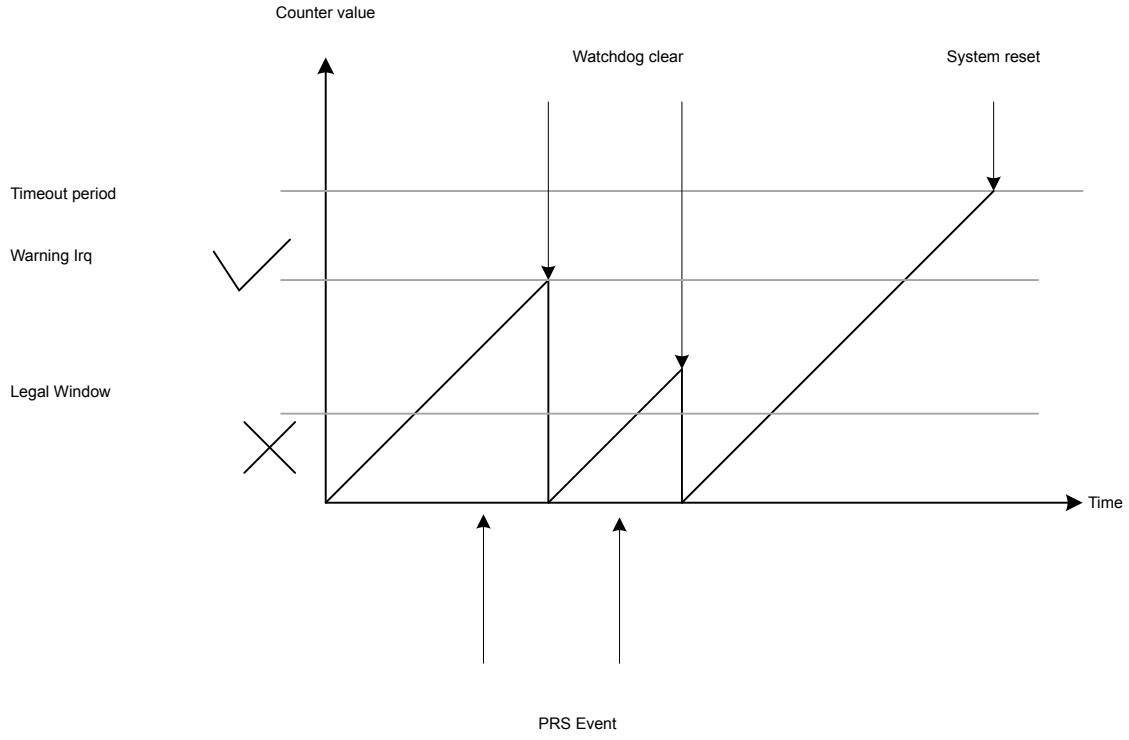


Figure 27.2. WDOG Warning, Window, and Timeout

27.3.6 PRS as Watchdog Clear

A PRS channel (selected by register PRS_CONSUMER_WDOGn_SRC0) can be used to clear the watchdog counter. To enable this feature, CLRSRC must be set to 1. Figure 27.2 PRS Clearing WDOG on page 980 shows how the PRS channel takes over the WDOG clear function. Clearing the WDOG with the PRS is mutually exclusive of clearing the WDT by software.

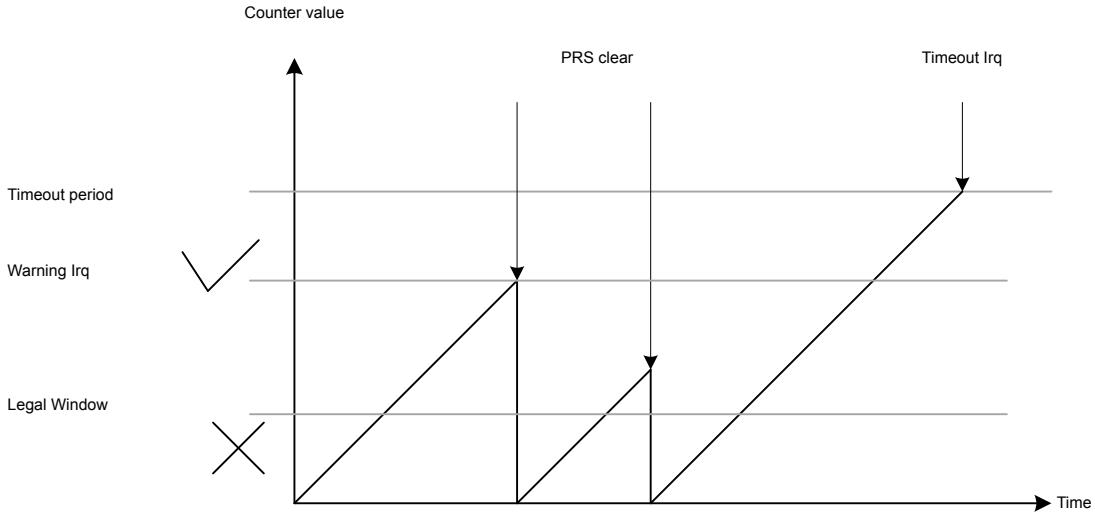


Figure 27.2. PRS Clearing WDOG

27.3.7 PRS Rising Edge Monitoring

PRS channels can be used to monitor multiple processes. The first and second channel are selected by PRS_CONSUMER_WDOGn_SRC0 and PRS_CONSUMER_WDOGn_SRC1, respectively. If enabled, every time the watch dog timer is cleared the PRS channels are checked and any channel which has not seen an event can trigger an interrupt.

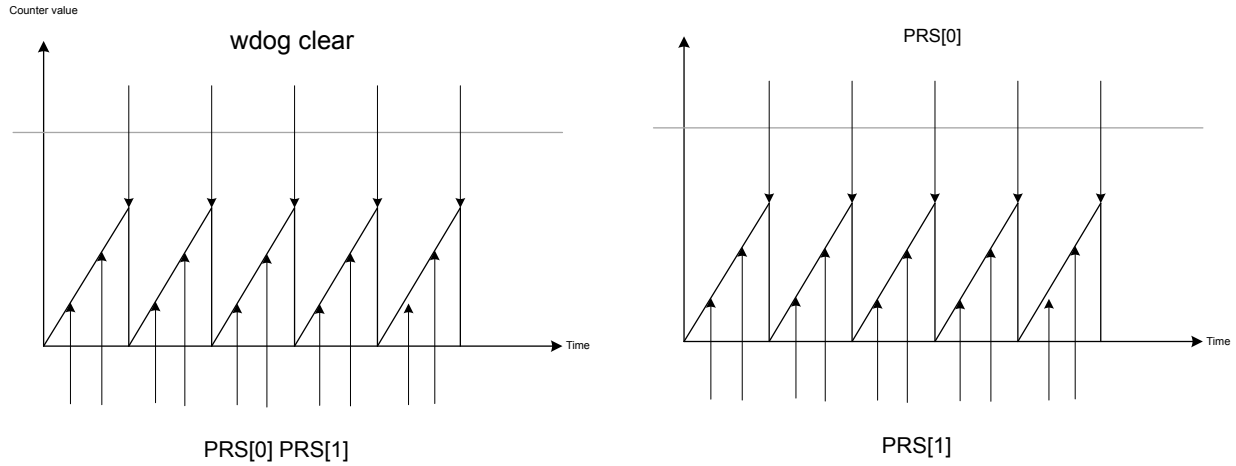


Figure 27.3. PRS Edge Monitoring in WDOG

27.4 WDOG Register Map

The offset register address is relative to the registers base address.

Offset	Name	Type	Description
0x000	WDOG_IPVERSION	R	IP Version Register
0x004	WDOG_EN	RW ENABLE	Enable Register
0x008	WDOG_CFG	RW CONFIG	Configuration Register
0x00C	WDOG_CMD	W LFSYNC	Command Register
0x014	WDOG_STATUS	RH	Status Register
0x018	WDOG_IF	RWH INTFLAG	Interrupt Flag Register
0x01C	WDOG_IEN	RW	Interrupt Enable Register
0x020	WDOG_LOCK	W	Lock Register
0x024	WDOG_SYNCBUSY	RH	Synchronization Busy Register
0x1000	WDOG_IPVERSION_SET	R	IP Version Register
0x1004	WDOG_EN_SET	RW ENABLE	Enable Register
0x1008	WDOG_CFG_SET	RW CONFIG	Configuration Register
0x100C	WDOG_CMD_SET	W LFSYNC	Command Register
0x1014	WDOG_STATUS_SET	RH	Status Register
0x1018	WDOG_IF_SET	RWH INTFLAG	Interrupt Flag Register
0x101C	WDOG_IEN_SET	RW	Interrupt Enable Register
0x1020	WDOG_LOCK_SET	W	Lock Register
0x1024	WDOG_SYNCBUSY_SET	RH	Synchronization Busy Register
0x2000	WDOG_IPVERSION_CLR	R	IP Version Register
0x2004	WDOG_EN_CLR	RW ENABLE	Enable Register
0x2008	WDOG_CFG_CLR	RW CONFIG	Configuration Register
0x200C	WDOG_CMD_CLR	W LFSYNC	Command Register
0x2014	WDOG_STATUS_CLR	RH	Status Register
0x2018	WDOG_IF_CLR	RWH INTFLAG	Interrupt Flag Register
0x201C	WDOG_IEN_CLR	RW	Interrupt Enable Register
0x2020	WDOG_LOCK_CLR	W	Lock Register
0x2024	WDOG_SYNCBUSY_CLR	RH	Synchronization Busy Register
0x3000	WDOG_IPVERSION_TGL	R	IP Version Register
0x3004	WDOG_EN_TGL	RW ENABLE	Enable Register
0x3008	WDOG_CFG_TGL	RW CONFIG	Configuration Register
0x300C	WDOG_CMD_TGL	W LFSYNC	Command Register
0x3014	WDOG_STATUS_TGL	RH	Status Register
0x3018	WDOG_IF_TGL	RWH INTFLAG	Interrupt Flag Register
0x301C	WDOG_IEN_TGL	RW	Interrupt Enable Register
0x3020	WDOG_LOCK_TGL	W	Lock Register

Offset	Name	Type	Description
0x3024	WDOG_SYNCBUSY_TGL	RH	Synchronization Busy Register

27.5 WDOG Register Description

27.5.1 WDOG_IPVERSION - IP Version Register

Offset	Bit Position																															
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x0	R	**IP Version**
	The read only IPVERSION field gives the version for this module. There may be minor software changes required for modules with different values of IPVERSION.			

27.5.2 WDOG_EN - Enable Register

Offset	Bit Position																															
0x004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																RW
Name																																EN

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0x0	RW	**Module Enable**
	The ENABLE bit enables the module. Software should write to CONFIG type registers before setting the ENABLE bit. Software should write to SYNC type registers only after setting the ENABLE bit.			

27.5.3 WDOG_CFG - Configuration Register

Offset	Bit Position																													
0x008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0					0x0							0xF								0x0	0x0	0x0				0x0	0x0	0x0	0x0	0x0
Access		RW					RW							RW								RW	RW	RW				RW	RW	RW	RW	RW
Name		WINSEL					WARNSEL							PERSEL								PRS1MISSRSTEN	PRS0MISSRSTEN	WDOGRSTDIS				DEBUGRUN	EM4BLOCK	EM3RUN	EM2RUN	CLRSRC

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:28	WINSEL	0x0	RW	**WDOG Illegal Window Select**

Select WDOG illegal limit.

Value	Mode	Description
0	DIS	Disabled.
1	SEL1	Window timeout is 12.5% of the Timeout.
2	SEL2	Window timeout is 25% of the Timeout.
3	SEL3	Window timeout is 37.5% of the Timeout.
4	SEL4	Window timeout is 50% of the Timeout.
5	SEL5	Window timeout is 62.5% of the Timeout.
6	SEL6	Window timeout is 75.5% of the Timeout.
7	SEL7	Window timeout is 87.5% of the Timeout.

27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:24	WARNSEL	0x0	RW	**WDOG Warning Period Select**

Select WDOG warning timeout period.

Value	Mode	Description
0	DIS	Disable
1	SEL1	Warning timeout is 25% of the Timeout.
2	SEL2	Warning timeout is 50% of the Timeout.
3	SEL3	Warning timeout is 75% of the Timeout.

23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PERSEL	0xF	RW	**WDOG Timeout Period Select**

Select WDOG timeout period.

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	SEL0		Timeout period of 9 wdog cycles
	1	SEL1		Timeout period of 17 wdog cycles
	2	SEL2		Timeout period of 33 wdog cycles
	3	SEL3		Timeout period of 65 wdog cycles
	4	SEL4		Timeout period of 129 wdog cycles
	5	SEL5		Timeout period of 257 wdog cycles
	6	SEL6		Timeout period of 513 wdog cycles
	7	SEL7		Timeout period of 1k wdog cycles
	8	SEL8		Timeout period of 2k wdog cycles
	9	SEL9		Timeout period of 4k wdog cycles
	10	SEL10		Timeout period of 8k wdog cycles
	11	SEL11		Timeout period of 16k wdog cycles
	12	SEL12		Timeout period of 32k wdog cycles
	13	SEL13		Timeout period of 64k wdog cycles
	14	SEL14		Timeout period of 128k wdog cycles
	15	SEL15		Timeout period of 256k wdog cycles
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	PRS1MISSRSTEN	0x0	RW	**PRS Src1 Missing Event WDOG Reset**
	When set, a PRS Source 1 missing event will trigger a WDOG reset.			
9	PRS0MISSRSTEN	0x0	RW	**PRS Src0 Missing Event WDOG Reset**
	When set, a PRS Source 0 missing event will trigger a WDOG reset.			
8	WDOGRSTDIS	0x0	RW	**WDOG Reset Disable**
	Disable WDOG reset output.			
	Value	Mode		Description
	0	EN		A timeout will cause a WDOG reset
	1	DIS		A timeout will not cause a WDOG reset
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	DEBUGRUN	0x0	RW	**Debug Mode Run**
	Set to keep WDOG running in debug mode.			
	Value	Mode		Description
	0	DISABLE		WDOG timer is frozen in debug mode
	1	ENABLE		WDOG timer is running in debug mode
3	EM4BLOCK	0x0	RW	**EM4 Block**

Bit	Name	Reset	Access	Description
	Set to disallow EM4 entry by software.			
	Value	Mode		Description
	0	DISABLE		EM4 can be entered by software. See EMU for detailed description.
	1	ENABLE		EM4 cannot be entered by software.
2	EM3RUN	0x0	RW	**EM3 Run**
	Set to keep WDOG running in EM3.			
	Value	Mode		Description
	0	DISABLE		WDOG timer is frozen in EM3.
	1	ENABLE		WDOG timer is running in EM3.
1	EM2RUN	0x0	RW	**EM2 Run**
	Set to keep WDOG running in EM2.			
	Value	Mode		Description
	0	DISABLE		WDOG timer is frozen in EM2.
	1	ENABLE		WDOG timer is running in EM2.
0	CLRSRC	0x0	RW	**WDOG Clear Source**
	Select WDOG clear source.			
	Value	Mode		Description
	0	SW		A write to the clear bit will clear the WDOG counter
	1	PRSSRC0		A rising edge on the PRS Source 0 will clear the WDOG counter

27.5.4 WDOG_CMD - Command Register

Offset	Bit Position																														
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																W
Name																																CLEAR

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CLEAR	0x0	W	**WDOG Timer Clear**

Clear WDOG timer. The bit must be written 4 WDOG cycles before the timeout.

Value	Mode	Description
0	UNCHANGED	WDOG timer is unchanged.
1	CLEARED	WDOG timer is cleared to 0.

27.5.5 WDOG_STATUS - Status Register

Offset	Bit Position																														
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	LOCK																															

Bit	Name	Reset	Access	Description
31	LOCK	0x0	R	**WDOG Configuration Lock Status**

Status of all lockable WDOG registers.

Value	Mode	Description
0	UNLOCKED	All WDOG lockable registers are unlocked.
1	LOCKED	All WDOG lockable registers are locked.

30:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

27.5.6 WDOG_IF - Interrupt Flag Register

Offset	Bit Position																														
0x018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												RW	RW	RW	RW	RW
Name																												PEM1	PEM0	WIN	WARN	TOUT

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	PEM1	0x0	RW	**PRS Src1 Event Missing Interrupt Flag**
	Set when a WDOG clear happens before a prs event has been detected on PRS Source one.			
3	PEM0	0x0	RW	**PRS Src0 Event Missing Interrupt Flag**
	Set when a WDOG clear happens before a prs event has been detected on PRS Source zero.			
2	WIN	0x0	RW	**WDOG Window Interrupt Flag**
	Set when a WDOG clear happens below the window limit value.			
1	WARN	0x0	RW	**WDOG Warning Timeout Interrupt Flag**
	Set when a WDOG warning timeout has occurred.			
0	TOUT	0x0	RW	**WDOG Timeout Interrupt Flag**
	Set when a WDOG timeout has occurred.			

27.5.7 WDOG_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x01C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																												0x0	0x0	0x0	0x0	0x0
Access																												RW	RW	RW	RW	RW
Name																												PEM1	PEM0	WIN	WARN	TOUT

Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	PEM1	0x0	RW	**PRS Src1 Event Missing Interrupt Enable**
	Enable/disable the PEM1 interrupt.			
3	PEM0	0x0	RW	**PRS Src0 Event Missing Interrupt Enable**
	Enable/disable the PEM0 interrupt.			
2	WIN	0x0	RW	**WDOG Window Interrupt Enable**
	Enable/disable the WIN interrupt.			
1	WARN	0x0	RW	**WDOG Warning Timeout Interrupt Enable**
	Enable/disable the WARN interrupt.			
0	TOUT	0x0	RW	**WDOG Timeout Interrupt Enable**
	Enable/disable the TOUT interrupt.			

27.5.8 WDOG_LOCK - Lock Register

Offset	Bit Position																															
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0xABE8															
Access																	W															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0xABE8	W	**WDOG Configuration Lock**

Write any other value than the unlock code to lock WDOG_EN, WDOD_CFG registers from editing. Write the unlock code to unlock.

Value	Mode	Description
0	LOCK	Lock WDOG lockable registers
44008	UNLOCK	Unlock WDOG lockable registers

27.5.9 WDOG_SYNCBUSY - Synchronization Busy Register

Offset	Bit Position																															
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0x0
Access																																R
Name																																CMD

Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CMD	0x0	R	**Sync Busy for Cmd Register**

CMD bitfield sync is busy when set.

28. Revision History

Revision 1.0

August, 2020

- Added register interface name to register chapter titles throughout document for better navigation.
- HFXO: added mention of clipped sine wave support.
- 3. System Processor: Corrected cycles for divide instruction.
- 6. MSC - Memory System Controller : Removed mention of "devices supporting read-while-write" from MSC_WRITECMD_ERASEMAIN0.
- 8.3.1.5 RHCLK - AHB Radio Clock: Changed text to clarify that this refers to the Radio AHB clock instead of the radio transicever clock fed directly by the crystal.
- Removed references to ACMP peripheral from EMU, EUART, and USART chapters. ACMP is not available on this device family.
- 10. SMU - Security Management Unit: Corrected register bit definitions: Added TIMER4 to PPUPATD0, PPUSATD0, and removed BUFC from PPUPATD1, PPUSATD1, BMPUPATD0, BMPUSATD0. Remaining bits moved to correct locations.
- 19. TIMER - Timer/Counter: Corrected equations for frequency calculations, to properly include prescaler.
- 21. USART - Universal Synchronous Asynchronous Receiver/Transmitter: Removed IRPRSEN bit from USART_IRCTRL register. Function does not exist in this device family.
- 25. GPIO - General Purpose Input/Output: Removed mention of THMSW_HALFSWITCH from Fixed Pin Function Table.
- 25. GPIO - General Purpose Input/Output: Removed registers and bit enumerations with no customer usage from descriptions.
- 25. GPIO - General Purpose Input/Output: Corrected register names for TIMERx CDTI0/1/2 signal routing.

Revision 0.5

March, 2020

- 3. System Processor: Corrected Cortex-M33 revision number.
- 4.2.3 SRAM: Added note about root code usage of RAM during a system reset.
- 4.2.1.2 Bus Faults: Removed sentence stating that the radio subsystem is the only peripheral which can generate a bus fault with clock disabled.
- 6. MSC - Memory System Controller : Corrected size of User Data space.
- Updated register descriptions for formatting conventions, typos, and clarity, and added missing register details throughout document.
- 8.3.3 RC Oscillator Calibration: Added names of relevant PRS consumer outputs.
- 11. CRYPTOACC - Cryptographic Accelerator: Removed mention of P-224 (unsupported), added HW_CFG registers.
- Added section: 12.3.1.3 EM1P .
- Updated information about debugging through EM2 and EM3 using the EM2DBGEN bit in the EMU2/3 and debugging sections.
- 12.3.7 Power Configurations: Updated with latest power configuration details.
- Added section: 12.3.9 EFP Communication.
- Updated 12.3.12 Temperature Sensor and added section 12.3.12.1 Linearization, Offset Correction, and Calibration.
- 20. PDM - PDM Interface: Fixed sections referring to more than 2 channels.
- 21.3.2.17 Single Data-link with External Driver: Added note about CSSETUP usage in asynchronous mode with AUTOCS.
- 22. EUART - Enhanced Universal Asynchronous Receiver/Transmitter: Removed LIN reference from Auto Baud Detection paragraph.
- 24.3.5.2 Internal and Dedicated Inputs: Corrected DECOUPLE voltage at positive input.
- Added section: 24.3.7.2 Output Polarity.
- Added section: 24.3.7.4 Output Resolution.
- Removed erroneous PADREFPOS and PADREFNEG options from IADC SINGLE and SCAN input selection registers.

Revision 0.1

July, 2019

Initial Release.

Appendix 1. Abbreviations

This section lists abbreviations used in this document.

Table 1.1. Abbreviations

Abbreviation	Description
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
AFC	Automatic Frequency Control
AGC	Automatic Gain Control
AHB	AMBA Advanced High-performance Bus. AMBA is short for "Advanced Microcontroller Bus Architecture".
APB	AMBA Advanced Peripheral Bus. AMBA is short for "Advanced Microcontroller Bus Architecture".
APC	Automatic Power Control
ASK	Amplitude Shift Keying
BLE	Bluetooth Low Energy
BLE-LR	Bluetooth Low Energy Long Range
BR	Baud Rate
BT	Bandwidth Time product
BUFC	Buffer Controller
BW	Bandwidth
CBC	Cipher Block Chaining (AES mode of operation)
CBC-MAC	Cipher Block Chaining - Message Authentication Code (AES mode of operation)
CC	Compare / Capture
CCA	Clear Channel Assessment
CFB	Cipher Feedback (AES mode of operation)
CHF	Channel Filter
CLK	Clock
CM3	ARM Cortex-M3
CM4	ARM Cortex-M4
CMD	Command
CMU	Clock Management Unit
CRC	Cyclic Redundancy Check
CTR	Counter mode (AES mode of operation)
CTRL	Control
DBG	Debug
DC	Direct Current
DEC	Decimator
DEMOD	Demodulator

Abbreviation	Description
DSA	Detection of Signal Arrival
DSSS	Direct Sequence Spread Spectrum
ECB	Electronic Code Book (AES mode of operation)
EFM32	Energy Friendly Microcontroller
EFR32	Wireless Gecko
EM	Energy Mode
EMU	Energy Management Unit
FEC	Forward Error Correction
FIR	Finite Impulse Response
FRC	Frame Controller
FSK	Frequency Shift Keying
GFSK	Gaussian Frequency Shift Keying
GMSK	Gaussian Minimum Shift Keying
GPIO	General Purpose Input / Output
HFRCO	High Frequency RC Oscillator
HFXO	High Frequency Crystal Oscillator
HW	Hardware
Hz	Hertz
IF	Intermediate Frequency
ISR	Interrupt Service Routine
LFRCO	Low Frequency RC Oscillator
LFXO	Low Frequency Crystal Oscillator
LNA	Low Noise Amplifier
LO	Local Oscillator
MOD	Modulator
MODEM	Modulator and Demodulator
MSK	Minimum Shift Keying
NRZ	Non Return to Zero
NVIC	Nested Vector Interrupt Controller
OFB	Output Feedback Mode (AES mode of operation)
OOK	On Off Keying
OQPSK	Offset Quadrature Phase Shift Keying
OSR	Over-Sampling Ratio
PA	Power Amplifier
PD	Power Down
PHY	Physical Layer
PROTIMER	Protocol Timer

Abbreviation	Description
PRS	Peripheral Reflex System
PWM	Pulse Width Modulation
RAC	Radio Controller
RAM	Random Access Memory
RF	Radio Frequency
RMU	Reset Management Unit
RSM	Radio State Machine
RSSI	Received Signal Strength Indicator
RTC	Real Time Counter
RX	Receive
SEQ	Radio Sequencer
SPI	Serial Peripheral Interface
SRC	Sample Rate Converter
STIMER	Sequencer Timer
SW	Software
SYNTH	Synthesizer
TX	Transmit
XTAL	Crystal

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com