EFM32 Giant Gecko 12 Family Reference Manual

The EFM32 Giant Gecko Series 1 MCUs are the world's most energy-friendly microcontrollers, featuring new connectivity interfaces and user interface features.

EFM32GG12 includes a powerful 32-bit ARM® Cortex®-M4 and provides robust security via a unique cryptographic hardware engine supporting AES, ECC, SHA, and True Random Number Generator (TRNG). New features include an SD/MMC/SDIO controller, Octal/Quad-SPI memory controller, CAN bus controller, PDM interface, highly robust capacitive sensing, enhanced alpha blending graphics engine, and LESENSE/PCNT enhancements for smart energy meters. These features, combined with ultra-low current active mode and short wake-up time from energy-saving modes, make EFM32GG12 microcontrollers well suited for any battery-powered application, as well as other systems requiring high performance and low-energy consumption.

Example applications:

- Smart energy meters
- Industrial and factory automation
- Home automation and security
- Mid- and high-tier wearables
- IoT devices

ENERGY FRIENDLY FEATURES

- ARM Cortex-M4 at 72 MHz
- Ultra low energy operation in active and sleep modes
- Octal/Quad-SPI memory interface w/ XIP
- SD/MMC/SDIO Host Controller
- PDM Microphone/Sensor Interface
- Dual CAN 2.0 Bus Controller
- Crystal-free low-energy USB
- Hardware cryptographic engine supports AES, ECC, SHA, and TRNG
- Robust capacitive touch sense
- Footprint compatible with select EFM32 packages
- 5 V tolerant I/O

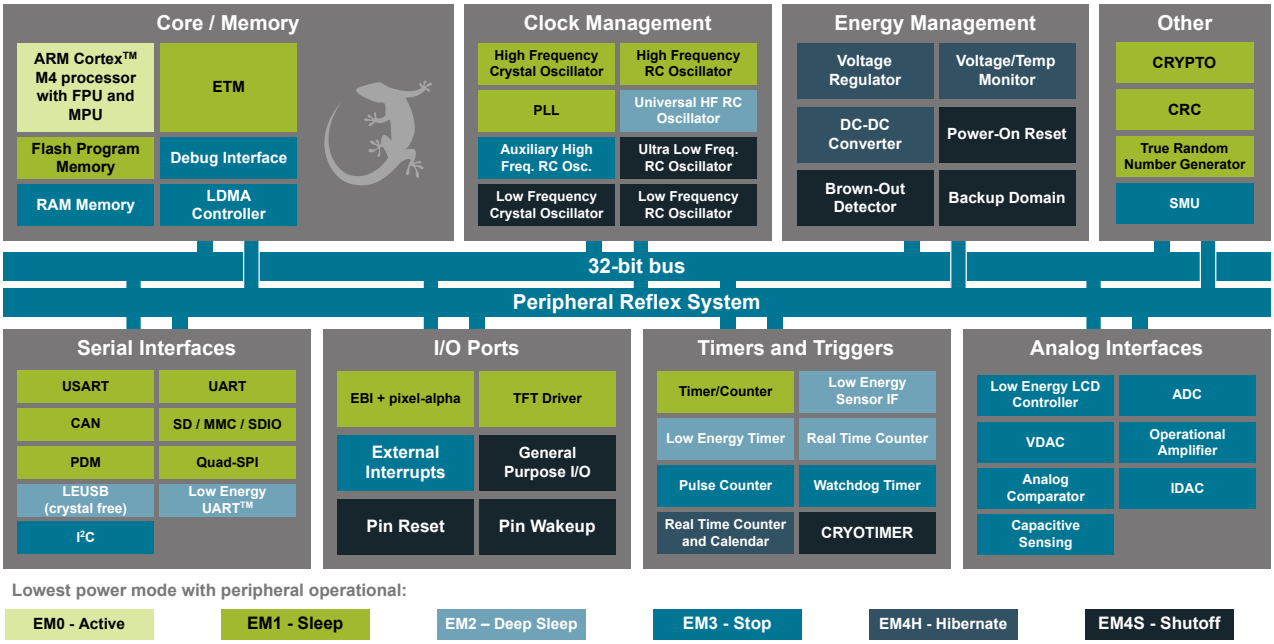


Table of Contents

1. About This Document

1.1 Introduction

This document contains reference material for the EFM32 Giant Gecko 12 devices. All modules and peripherals in the EFM32 Giant Gecko 12 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 and applicable errata documents.

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 4.2 System Address Space With Core and Code Space Listing on page 54), 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 39.

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 IFC registers (see 3.3.1.2 IFC Read-clear 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 4.2.3 Peripheral Bit Set and Clear)

Access Type	Description
RW(a), R(a), etc.	"(a)" suffix indicates that register has actionable reads (see 5.3.7 Debugger Reads of Actionable Registers)

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 data sheet.

1.3 Related Documentation

Further documentation on the EFM32 Giant Gecko 12 devices and the ARM Cortex-M4 can be found at the Silicon Labs and ARM web pages:

www.silabs.com

www.arm.com

2. System Overview

2.1 Introduction

The EFM32 MCUs are the world's most energy friendly microcontrollers. With a unique combination of the powerful 32-bit ARM Cortex-M4, innovative low energy techniques, short wake-up time from energy saving modes, and a wide selection of peripherals, the EFM32 Giant Gecko 12 microcontroller is well suited for any battery operated application as well as other systems requiring high performance and low-energy consumption.

2.2 Features

- **ARM Cortex-M4 CPU platform**
  - High Performance 32-bit processor @ up to 72 MHz
  - DSP instruction support and Floating Point Unit
  - Memory Protection Unit
  - Wake-up Interrupt Controller
- **Flexible Energy Management System**
  - Power routing configurations including integrated DCDC converter
  - Voltage Monitoring and Brown Out Detection
  - State Retention
- **Up to 1024 KB Flash**
  - Dual-bank with read-while-write support
- **Octal/Quad-SPI Flash Memory Interface**
  - Supports 3 V and 1.8 V memories
  - 1/2/4/8-bit data bus
  - Quad-SPI Execute In Place (XIP)
  - Up to 30 MHz SDR/DDR at 3 V
- **192 KB RAM**
  - Includes ECC (SEC-DED)
- **Up to 95 General Purpose I/O Pins**
  - Configurable push-pull, open-drain, pull-up/down, input filter, drive strength
  - Configurable peripheral I/O locations
  - 5 V tolerance on select pins
  - Asynchronous external interrupts
  - Output state retention and wake-up from Shutoff Mode
- **12 Channel DMA Controller**
  - Alternate/primary descriptors with scatter-gather/ping-pong operation
- **Up to 16 Channel Peripheral Reflex System**
  - Enables autonomous inter-peripheral signaling while core is asleep
- **External Bus Interface**
  - Up to 4x256 MB of external memory mapped space
  - TFT Controller supporting Direct Drive
  - Per-pixel alpha-blending engine
- **Low-energy Universal Serial Bus (USB) with Device, Host, and OTG support**
  - Fully USB 2.0 compliant
  - On-chip PHY and embedded 5V to 3.3V regulator
  - Crystal-free Device mode operation
  - Patent-pending Low-Energy Mode (LEM)

- **Integrated LCD Controller**
  - Up to 4 x 40 segments or 8 x 36 segments in octaplex mode
  - Voltage boost, contrast and autonomous animation
  - Patented low-energy LCD driver
- **CRYPTO Advanced Encryption Standard Accelerator**
  - AES encryption / decryption, with 128 or 256 bit keys
  - Multiple AES modes of operation, including Counter (CTR), Galois/Counter Mode (GCM), Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB).
  - Accelerated SHA-1 and SHA-2
  - Accelerated Elliptic Curve Cryptography (ECC), with binary or prime fields
  - Flexible 256-bit ALU and sequencer
- **True Random Number Generator (TRNG)**
- **Hardware CRC engine**
  - Single-cycle computation with 8/16/32-bit data and 16-bit (programmable)/32-bit (fixed) polynomial
- **Security Management Unit (SMU)**
  - Fine-grained access control for on-chip peripherals
- **Additional Communication interfaces**
  - SD/MMC/SDIO Host Controller
    - SD v3.01, SDIO v3.0 and MMC v4.51 up to 50 MHz
    - 1/4/8-bit bus width
  - Up to 2 × CAN Bus Controller
    - Version 2.0A and 2.0B up to 1 Mbps
  - Up to 5 x Universal Synchronous/Asynchronous Receiver/Transmitter
    - UART/SPI/SmartCard (ISO 7816)/IrDA/I2S/LIN
    - Triple buffered full/half-duplex operation
    - Hardware flow control
    - 4-16 data bits
    - USART2 available on high-speed clock domain
  - 2 x Universal Asynchronous Receiver/Transmitter
    - Triple buffered full/half-duplex operation
    - 8-9 data bits
  - 2 x Low Energy UART
    - Autonomous operation with DMA in Deep Sleep Mode
  - 2 x I^2C Interface with SMBus support
    - Address recognition in EM3 Stop Mode
  - 4-channel PDM Interface
    - Supports microphone and sensor applications

- **Timers/Counters**
  - 2 x 32-bit Timer/Counter
    - 3/4 Compare/Capture/PWM channels
    - Dead-Time Insertion
  - 4 x 16-bit Timer/Counter
    - 3 Compare/Capture/PWM channels
    - Dead-Time Insertion
  - 2 x 16-bit Low Energy Timer
  - 32-bit Ultra Low Energy Timer/Counter (CRYOTIMER) for periodic wake-up from any Energy Mode
  - 24-bit Real-Time Counter (RTC)
  - 32-bit Real-Time Counter and Calendar (RTCC)
  - 3 x 16-bit Pulse Counter
    - Asynchronous pulse counting/quadrature decoding
  - 2 x Watchdog Timer
- **Backup Power Domain**
  - RTCC and retention registers in a separate power domain, available in all energy modes
  - Operation from backup battery when main power absent/insufficient
- **Ultra low power precision analog peripherals**
  - Up to 83 GPIO pins are analog-capable. Flexible analog peripheral-to-pin routing via Analog Port (APORT)
  - 2 x 12-bit 1 Msamples/s Analog to Digital Converter (ADC)
    - Single ended or differential operation
    - Conversion tailgating for predictable latency
    - On-chip temperature sensor
  - 12-bit 500 ksamples/s Digital to Analog Converter (VDAC)
    - 2 single ended channels/1 differential channel
  - Up to 4 Operational Amplifiers
    - Supports rail-to-rail inputs and outputs
    - Programmable gain
  - 3 x Analog Comparator (ACMP)
    - Programmable speed/current
    - Capacitive sensing with up to 8 inputs
  - Current Digital to Analog Converter (IDAC)
    - Source or sink a configurable constant current
  - Capacitive Sense Module (CSEN)
    - Robust current-based capacitive sensing with up to 64 inputs and wake-on-touch
  - Supply Voltage Monitor (VMON)
- **Ultra low power sensor interface (LESENSE)**
  - Autonomous sensor monitoring in Deep Sleep Mode
  - Wide range of sensors supported, including LC sensors and capacitive buttons
  - Up to 16 inputs
- **Ultra efficient Power-on Reset and Brown-Out Detector**
- **Debug Interface**
  - 2-pin Serial Wire Debug interface
  - 1-pin Serial Wire Viewer
  - 4-pin JTAG interface
  - Embedded Trace Macrocell (ETM)

2.3 Block Diagram

A block diagram of EFM32 Giant Gecko 12 is shown in Figure 2.1 Diagram of EFM32 Giant Gecko 12 on page 44. The color indicates peripheral availability in energy modes as described in 2.4 Energy Modes.

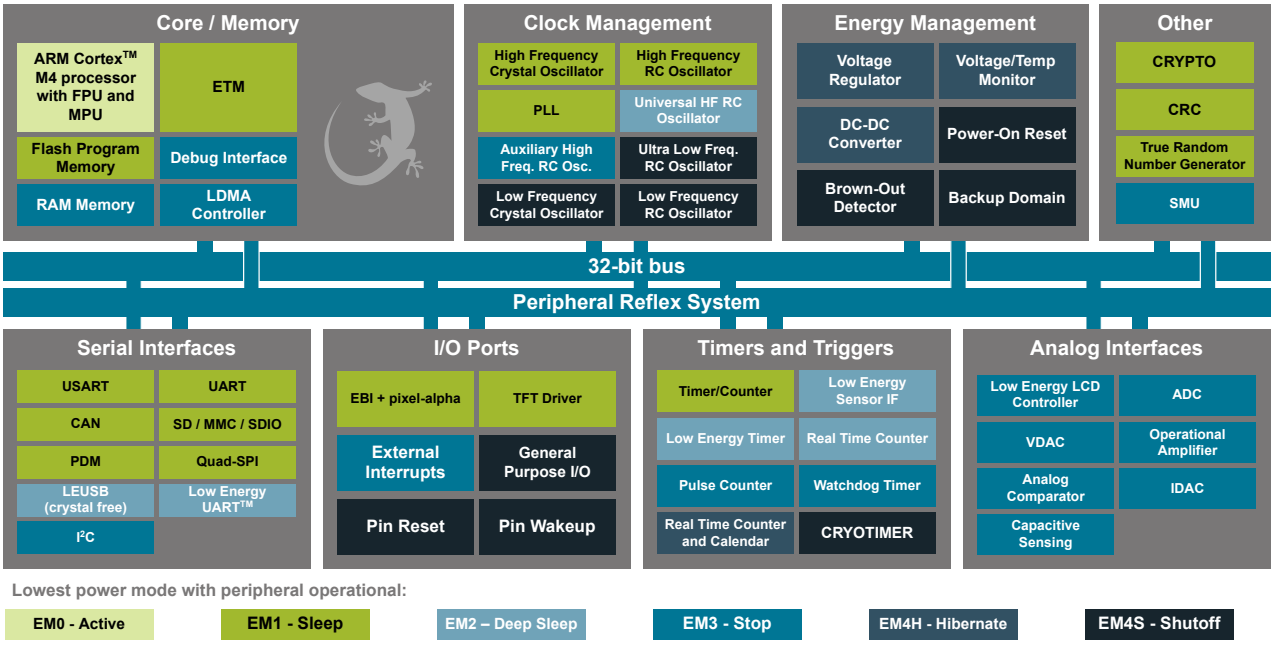


Figure 2.1. Diagram of EFM32 Giant Gecko 12

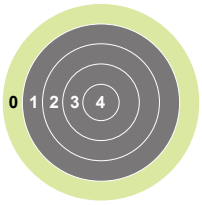
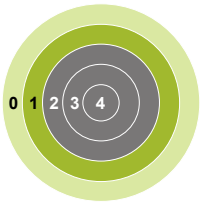
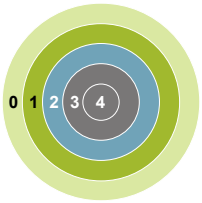
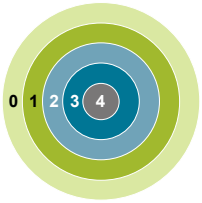
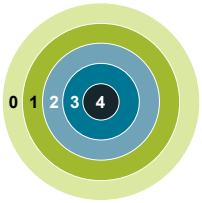
Note: In the block diagram, color indicates availability in different energy modes.

2.4 Energy Modes

There are five different Energy Modes (EM0 Active-EM4 Hibernate/Shutoff) in the EFM32 Giant Gecko 12, see Table 2.1 Energy Mode Description on page 45. The EFM32 Giant Gecko 12 is designed to achieve a high degree of autonomous operation in low energy modes. The intelligent combination of peripherals, RAM with data retention, DMA, low-power oscillators and short wake-up times, makes it attractive to remain in low energy modes for long periods and thus saving energy consumption.

Throughout this document, the first figure in every module description contains an Energy Mode Indicator that shows in which energy mode(s) the module can operate (see Table 2.1 Energy Mode Description on page 45).

Table 2.1. Energy Mode Description

Energy Mode	Name	Description
	EM0 Active – Energy Mode 0 (Run mode)	In EM0 Active, the CPU is actively running code. All peripherals can also be activated.
	EM1 Sleep – Energy Mode 1 (Sleep Mode)	In EM1 Sleep, the CPU is sleeping in a low-power state. All peripherals, including DMA, PRS, and the memory system are still available.
	EM2 DeepSleep – Energy Mode 2 (Deep Sleep Mode)	In EM2 DeepSleep the high frequency oscillator is turned off, but with the 32.768 kHz oscillator running, selected low energy peripherals (LCD, RTC, LETIMER, PCNT, WDOG, LEUART, I^2C, ACMP, LESENSE , USB) are still available, giving a high degree of autonomous operation with very low current consumption and fast wake times. Power-on Reset, Brown-out Detection and full RAM and CPU retention is also included.
	EM3 Stop - Energy Mode 3 (Stop Mode)	In EM3 Stop the low-frequency oscillator is disabled, but there is still full CPU and RAM retention, as well as Power-on Reset, Pin reset EM4 Hibernate/Shutoff wake-up and Brown-out Detector, with very low current consumption. The low-power ACMP, asynchronous external interrupt, PCNT, and I^2C can quickly wake the device.
	EM4 Hibernate/Shutoff – Energy Mode 4 (Shutoff Mode)	In EM4 Hibernate/Shutoff, the current is extremely low and all chip functionality is turned off except the pin reset , GPIO pin wake-up , GPIO pin retention and the power on reset. All pins are put into their reset state.

2.5 Timers

EFM32 Giant Gecko 12 includes multiple timers, as can be seen from Table 2.2 EFM32 Giant Gecko 12 Timers Overview on page 46.

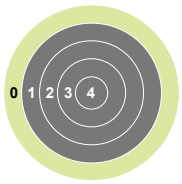
Table 2.2. EFM32 Giant Gecko 12 Timers Overview

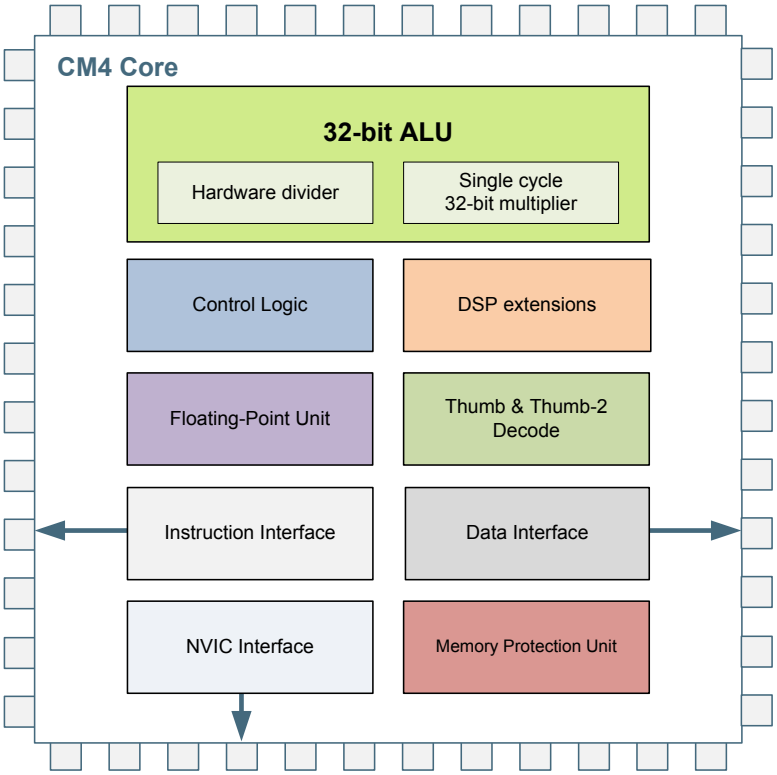
Timer	Number of instances	Typical clock source	Overview
RTCC	1	Low frequency (LFXO or LFRCO)	32 bit Real Time Counter and Calendar, typically used to accurately time inactive periods and enable wakeup on compare match.
RTC	1	Low frequency (LFXO or LFRCO)	24 bit Real Time Counter, typically used to accurately time inactive periods and enable wakeup on compare match.
TIMER	4	High frequency (HFXO or HFRCO)	16 bit general purpose timer.
WTIMER	2	High frequency (HFXO or HFRCO)	32 bit general purpose timer.
SysTick timer	1	High frequency (HFXO or HFRCO) or low frequency (LFXO, LFRCO or ULFRCO)	32 bit SysTick timer integrated in the Cortex-M4. Typically used as an Operating System timer.
WDOG	2	Low frequency (LFXO, LFRCO or ULFRCO)	Watch dog timer. Once enabled, this module must be periodically accessed. If not, this is considered an error and the EFM32 Giant Gecko 12 is reset in order to recover the system.
LETIMER	2	Low frequency (LFXO, LFRCO or ULFRCO)	Low energy general purpose timer.
PCNT	3	Low frequency (LFXO, LFRCO or ULFRCO) or external pin	Low energy pulse counter with quadrature mode.
CRYOTIMER	1	Low frequency (LFXO, LFRCO or ULFRCO)	Ultra Low energy 32 bit timer available in all Energy Modes

Advanced interconnect features allows synchronization between timers. This includes:

- Start / stop any high frequency timer synchronized with the RTCC

3. System Processor





Quick Facts

What?

The industry leading Cortex-M4 processor from ARM is the CPU in the EFM32 Giant Gecko 12 devices.

Why?

The ARM Cortex-M4 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 EFM32 Giant Gecko 12 devices, the Cortex-M4 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-M4 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-M4 implemented is revision r0p1.

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 12 cycles
- Atomic bit manipulation with bit banding
  - Direct access to single bits of data
  - Two 1MB bit banding regions for memory and peripherals mapping to 32MB alias regions
  - Atomic operation, cannot be interrupted by other bus activities
- 1.25 DMIPS/MHz
- Memory Protection Unit
  - Up to 8 protected memory regions
- 24 bits 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
- Alligned 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-M4 implementation in the EFM32 Giant Gecko 12 family, the reader is referred to the ARM Cortex-M4 documentation provided by ARM.

3.3.1 Interrupt Operation

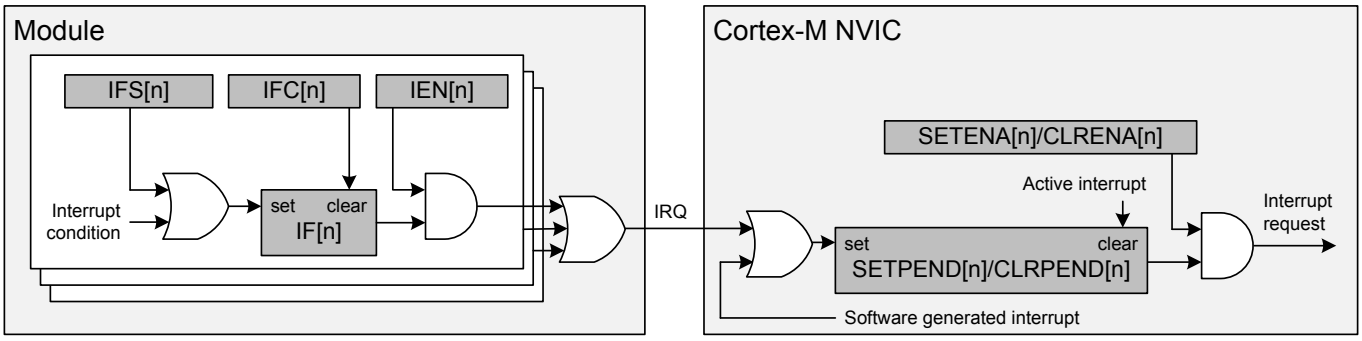


Figure 3.1. Interrupt Operation

The interrupt request (IRQ) lines are connected to the Cortex-M4. Each of these lines (shown in Table 3.1 Interrupt Request Lines (IRQ) on page 50) 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 IFS/IFC registers. 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 ISPR0/ICPR0) in the Cortex-M4 NVIC. The pending bit is then qualified with an enable bit (set/cleared with SETENA/CLRENA bits in ISER0/ICER0) before generating an interrupt request to the core. Figure 3.1 Interrupt Operation on page 49 illustrates the interrupt system. For more information on how the interrupts are handled inside the Cortex-M4, the reader is referred to the **ARM Cortex-M4 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());
}
```

3.3.1.2 IFC Read-clear Operation

In addition to the normal interrupt setting and clearing operations via the IFS/IFC registers, there is an additional atomic Read-clear operation that can be enabled by setting IFCREADCLEAR=1 in the MSC_CTRL register. When enabled, reads of peripheral IFC registers will return the interrupt vector (mirroring the IF register), while at the same time clearing whichever interrupt flags are set. This operation is functionally equivalent to reading the IF register and then writing the result immediately back to the IFC register.

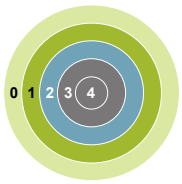
3.3.2 Interrupt Request Lines (IRQ)

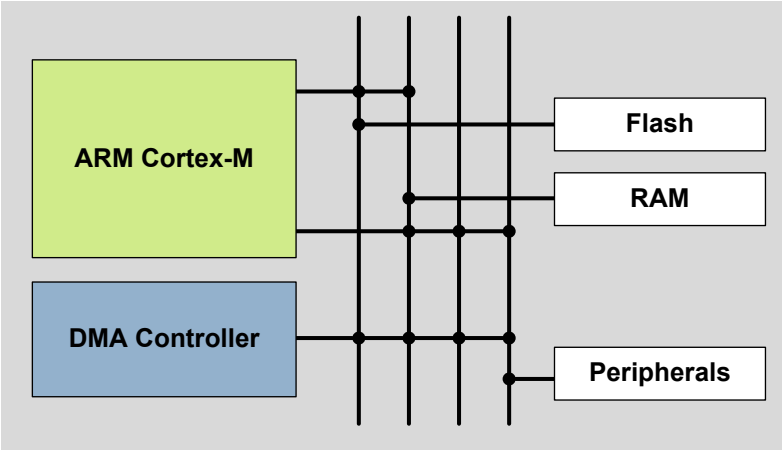
Table 3.1. Interrupt Request Lines (IRQ)

IRQ #	Source(s)
0	EMU
1	WDOG0
2	LDMA
3	GPIO_EVEN
4	SMU
5	TIMER0
6	USART0_RX
7	USART0_TX
8	ACMP0 ACMP1
9	ADC0
10	IDAC0
11	I2C0
12	I2C1
13	GPIO_ODD
14	TIMER1
15	TIMER2
16	TIMER3
17	USART1_RX
18	USART1_TX
19	USART2_RX
20	USART2_TX
21	UART0_RX
22	UART0_TX
23	UART1_RX
24	UART1_TX
25	LEUART0
26	LEUART1
27	LETIMER0
28	PCNT0
29	PCNT1
30	PCNT2
31	RTCC
32	CMU

IRQ #	Source(s)
33	MSC
34	CRYPTO0
35	CRYOTIMER
36	FPUEH
37	USART3_RX
38	USART3_TX
39	USART4_RX
40	USART4_TX
41	WTIMER0
42	WTIMER1
43	VDAC0
44	CSEN
45	LESENSE
46	EBI
47	ACMP2
48	ADC1
49	LCD
50	SDIO
51	CAN0
52	CAN1
53	USB
54	RTC
55	WDOG1
56	LETIMER1
57	TRNG0
58	QSPI0
59	PDM

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 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 DeepSleep and EM3 Stop.

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 Active to EM3 Stop removes the data restore time penalty, and the DMA ensures fast autonomous transfers with predictable response time.

4.1 Introduction

The EFM32 Giant Gecko 12 contains an AMBA AHB Bus system to allow bus masters to access the memory mapped address space. A multilayer AHB bus matrix connects the 6 master bus interfaces to the AHB slaves (Figure 4.1 EFM32 Giant Gecko 12 Bus System on page 53). 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 6 AHB bus masters are:

- **Cortex-M4 ICode:** Used for instruction fetches from Code memory (valid address range: 0x00000000 - 0x1FFFFFFF)
- **Cortex-M4 DCode:** Used for debug and data access to Code memory (valid address range: 0x00000000 - 0x1FFFFFFF)
- **Cortex-M4 System:** Used for data and debug access to system space. It can access entire memory space except Code memory (valid address range: 0x20000000 - 0xFFFFFFFF)
- **DMA:** Can access the entire memory space except the internal core memory region, the EBI code region (0x12000000 - 0x1FFFFFFF), the QSPI code region (0x04000000 - 0x0BFFFFFF) and the DMEM code region
- **USB DMA:** Can access EBI (0x80000000 - 0xBFFFFFFF), QSPI (0xC0000000 - 0xCFFFFFFF), Flash (0x00000000 - 0x03FFFFFF), Flash Info (0x0F000000 - 0x0FFFFFFF) and RAM (0x20000000 - 0x3FFFFFFF).
- **SDIO DMA:** Can access EBI (0x80000000 - 0xBFFFFFFF), QSPI (0xC0000000 - 0xCFFFFFFF), Flash (0x00000000 - 0x03FFFFFF), Flash Info (0x0F000000 - 0x0FFFFFFF) and RAM (0x20000000 - 0x3FFFFFFF).

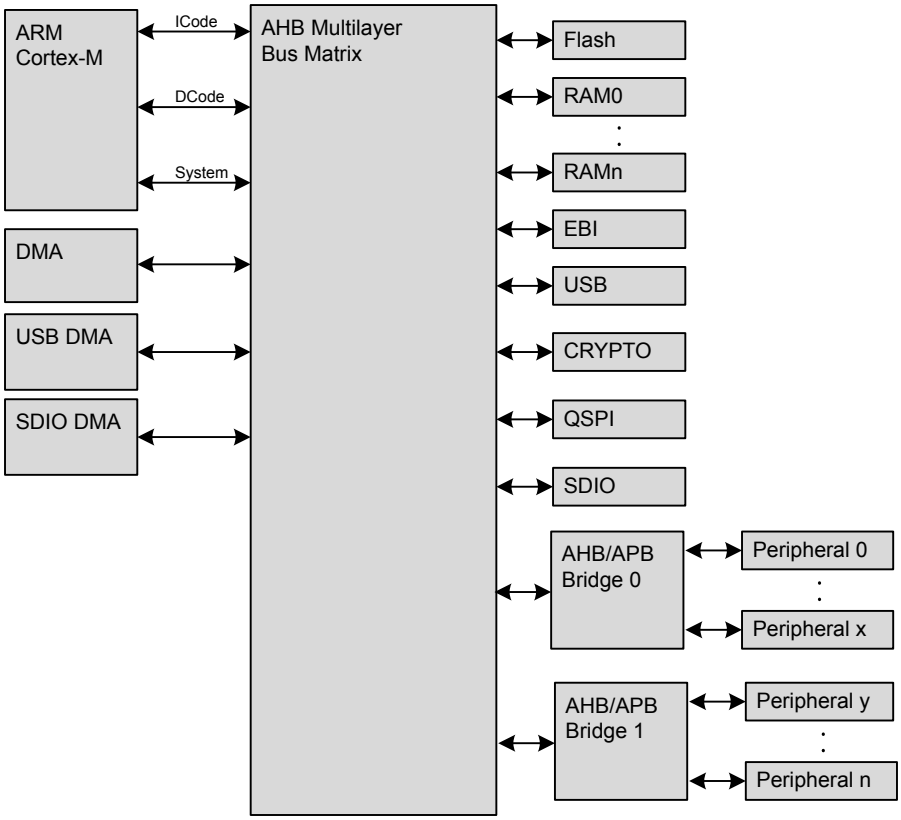


Figure 4.1. EFM32 Giant Gecko 12 Bus System

4.2 Functional Description

The memory segments are mapped together with the internal segments of the Cortex-M4 into the system memory map shown by Figure 4.2 System Address Space With Core and Code Space Listing on page 54.

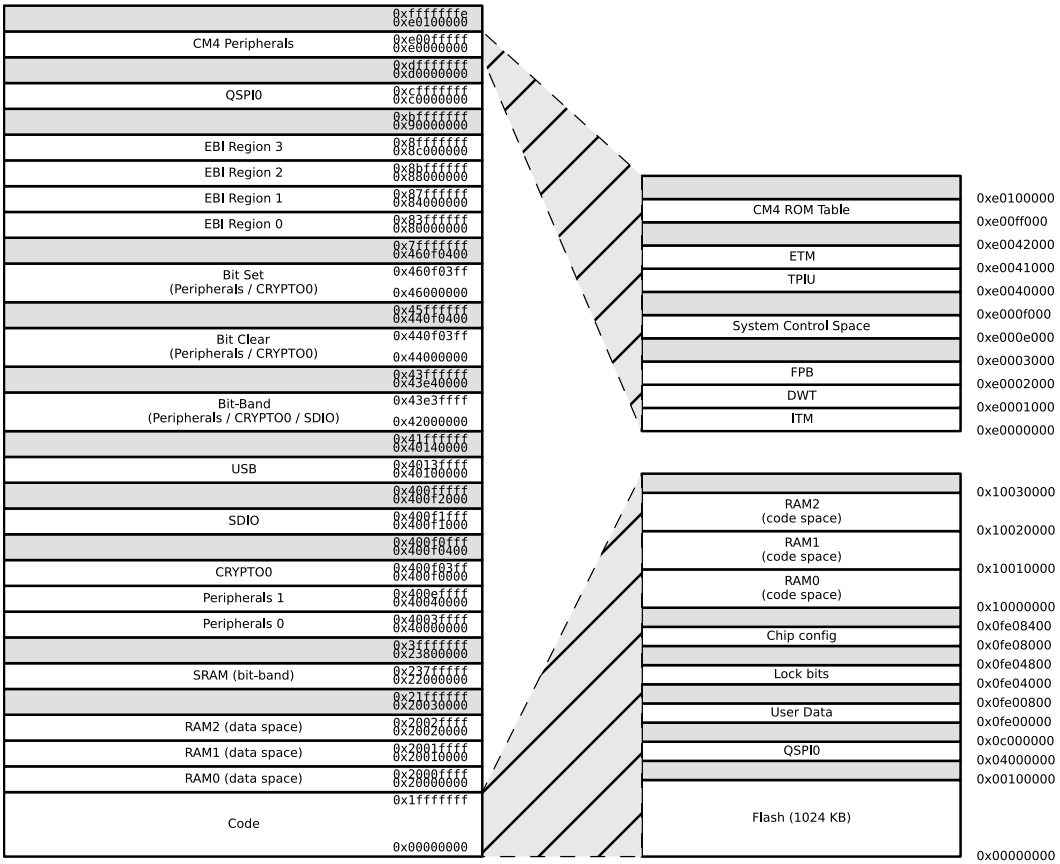


Figure 4.2. System Address Space With Core and Code Space Listing

Additionally, the peripheral address map is detailed by Figure 4.3 System Address Space With Peripheral Listing on page 55.

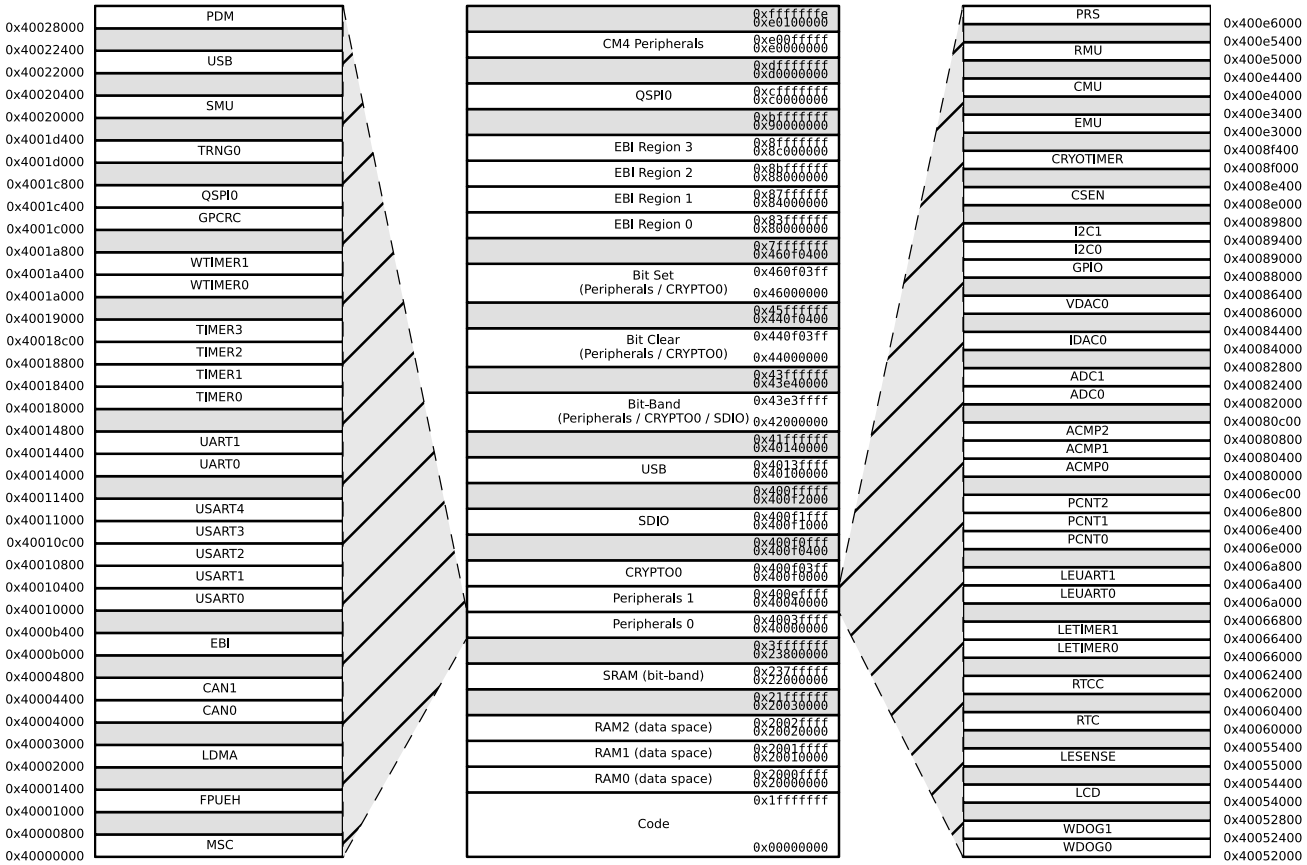


Figure 4.3. System Address Space With Peripheral Listing

The embedded SRAM is located at address 0x20000000 in the memory map of the EFM32 Giant Gecko 12. When running code located in SRAM starting at this address, the Cortex-M4 uses the System bus interface to fetch instructions. This results in reduced performance as the Cortex-M4 accesses stack, other data in SRAM and peripherals using the System bus interface. To be able to run code from SRAM efficiently, the SRAM is also mapped in the code space at address 0x10000000.

When running code from this space, the Cortex-M4 fetches instructions through the I/D-Code bus interface, leaving the System bus interface for data access.

The SRAM mapped into the code space can however only be accessed by the CPU and not any other bus masters, e.g. DMA. See 4.5 SRAM for more detailed info on the system SRAM.

4.2.1 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.

Note: The implementation of bit-banding in the core is such that bit-band accesses forward the transfer size info into the actual bus transfer size, so the same restrictions apply to bit-band accesses as apply to normal read/write accesses.

4.2.2 Bit-banding

The SRAM bit-band alias and peripheral bit-band alias regions are located at 0x22000000 and 0x42000000 respectively. Read and write operations to these regions are converted into masked single-bit reads and atomic single-bit writes to the embedded SRAM and peripherals of the EFM32 Giant Gecko 12.

Note: Bit-banding is only available through the CPU. No other AHB masters (e.g. DMA) can perform Bit-banding operations.

Using a standard approach to modify a single register or SRAM bit in the aliased regions, would require software to read the value of the byte, half-word or word containing the bit, modify the bit, and then write the byte, half-word or word back to the register or SRAM address. Using bit-banding, this can be done in a single operation, consuming only two bus cycles. As read-writeback, bit-masking and bit-shift operations are not necessary in software, code size is reduced and execution speed improved.

The bit-band regions allow each bit in the SRAM and Peripheral areas of the memory map to be addressed. To set or clear a bit in the embedded SRAM, write a 1 or a 0 to the following address:

$$bit\_address = 0x22000000 + (address - 0x20000000) \cdot 32 + bit \cdot 4$$

where address is the address of the 32-bit word containing the bit to modify, and bit is the index of the bit in the 32-bit word.

To modify a bit in the Peripheral area, use the following address:

$$bit\_address = 0x42000000 + (address - 0x40000000) \cdot 32 + bit \cdot 4$$

4.2.3 Peripheral Bit Set and Clear

The EFM32 Giant Gecko 12 supports bit set and bit clear access to all peripherals except those listed in Table 4.1 Peripherals that Do Not Support Bit Set and Bit Clear on page 57. The bit set and bit clear functionality (also called Bit Access) enables modification of bit fields (single bit or multiple bit wide) without the need to perform a read-modify-write (though it is functionally equivalent). Also, the operation is contained within a single bus access (for HF peripherals), unlike the Bit-banding operation described in section 4.2.2 Bit-banding which consumes two bus accesses per operation. All AHB masters except USB DMA, and SDIO DMA can utilize this feature.

The bit clear aliasing region starts at 0x44000000 and the bit set aliasing region starts at 0x46000000. Thus, to apply a bit set or clear operation, write the bit set or clear mask to the following addresses:

bit_clear_address = address + 0x04000000

bit_set_address = address + 0x06000000

For bit set operations, bit locations that are 1 in the bit mask will be set in the destination register:

register = (register OR mask)

For bit clear operations, bit locations that are 1 in the bit mask will be cleared in the destination register:

register = (register AND (NOT mask))

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 25 to 26, the field would initially take on a value of zero. If the analog module is active at the time, this could lead to undesired behavior.

The peripherals listed in Table 4.1 Peripherals that Do Not Support Bit Set and Bit Clear on page 57 do not support Bit Access for any registers. All other peripherals do support Bit Access, however, there may be cases of certain registers that do not support it. Such registers have a note regarding this lack of support.

Table 4.1. Peripherals that Do Not Support Bit Set and Bit Clear

Module
EMU
RMU
SDIO
CAN0
CAN1
QSPI0
CRYOTIMER
USB
TRNG0

4.2.4 Peripherals

The peripherals are mapped into the peripheral memory segment, each with a fixed size address range according to Table 4.2 Peripherals on page 58, Table 4.3 Low Energy Peripherals on page 59 , and Table 4.4 Core Peripherals on page 59.

Table 4.2. Peripherals

Address Range	Module Name
0x400F1000 - 0x400F2000	SDIO
0x400E6000 - 0x400E6400	PRS
0x4008F000 - 0x4008F400	CRYOTIMER
0x4008E000 - 0x4008E400	CSEN
0x40089400 - 0x40089800	I2C1
0x40089000 - 0x40089400	I2C0
0x40088000 - 0x40089000	GPIO
0x40086000 - 0x40086400	VDAC0
0x40084000 - 0x40084400	IDAC0
0x40082400 - 0x40082800	ADC1
0x40082000 - 0x40082400	ADC0
0x40080800 - 0x40080C00	ACMP2
0x40080400 - 0x40080800	ACMP1
0x40080000 - 0x40080400	ACMP0
0x40028000 - 0x40029000	PDM
0x40020000 - 0x40020400	SMU
0x4001D000 - 0x4001D400	TRNG0
0x4001C400 - 0x4001C800	QSPI0
0x4001C000 - 0x4001C400	GPCRC
0x4001A400 - 0x4001A800	WTIMER1
0x4001A000 - 0x4001A400	WTIMER0
0x40018C00 - 0x40019000	TIMER3
0x40018800 - 0x40018C00	TIMER2
0x40018400 - 0x40018800	TIMER1
0x40018000 - 0x40018400	TIMER0
0x40014400 - 0x40014800	UART1
0x40014000 - 0x40014400	UART0
0x40011000 - 0x40011400	USART4
0x40010C00 - 0x40011000	USART3
0x40010800 - 0x40010C00	USART2
0x40010400 - 0x40010800	USART1
0x40010000 - 0x40010400	USART0
0x40004400 - 0x40004800	CAN1

Address Range	Module Name
0x40004000 - 0x40004400	CAN0

Table 4.3. Low Energy Peripherals

Address Range	Module Name
0x4006E800 - 0x4006EC00	PCNT2
0x4006E400 - 0x4006E800	PCNT1
0x4006E000 - 0x4006E400	PCNT0
0x4006A400 - 0x4006A800	LEUART1
0x4006A000 - 0x4006A400	LEUART0
0x40066400 - 0x40066800	LETIMER1
0x40066000 - 0x40066400	LETIMER0
0x40062000 - 0x40062400	RTCC
0x40060000 - 0x40060400	RTC
0x40055000 - 0x40055400	LESENSE
0x40054000 - 0x40054400	LCD
0x40052400 - 0x40052800	WDOG1
0x40052000 - 0x40052400	WDOG0

Table 4.4. Core Peripherals

Address Range	Module Name
0xE0041000 - 0xE0081000	ETM
0xE0000000 - 0xE0040000	CM4
0x400F0000 - 0x400F0400	CRYPTO0
0x40022000 - 0x40022400	USB
0x4000B000 - 0x4000B400	EBI
0x40002000 - 0x40003000	LDMA
0x40001000 - 0x40001400	FPUEH
0x40000000 - 0x40000800	MSC

4.2.5 Bus Matrix

The Bus Matrix connects the memory segments to the bus masters as detailed in 4.1 Introduction.

4.2.5.1 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.

4.2.5.2 Peripheral Access Performance

The Bus Matrix is a multi-layer energy optimized AMBA AHB compliant bus with an internal bandwidth of 6x a single AHB interface.

The Cortex-M4, DMA Controller, and peripherals (not peripherals in the low frequency clock domain) run on clocks which can be pre-scaled separately. Clocks and prescaling are described in more detail in 10. CMU - Clock Management Unit . This section describes the expected bus wait states for a peripheral based on its frequency relative to the HFCLK frequency. For this discussion, PERCLK refers to a selected peripheral's clock frequency, which is some integer division of the HFCLK frequency.

Another factor that effects the cycle latency of peripheral accesses is the Peripheral Access Wait Mode (WAITMODE in MSC_CTRL) configuration. For instance, when set to WS0, a higher throughput (in terms of HFCLK cycles) is possible than with a higher wait state setting.

4.2.5.2.1 WS0 Mode

In general, when accessing a peripheral, the latency in number of HFCLK cycles, not including master arbitration, is given by:

$$N_{bus\ cycles} = N_{slave\ cycles} \cdot f_{HFCLK}/f_{PERCLK}, \text{ best-case write accesses}$$

$$N_{bus\ cycles} = N_{slave\ cycles} \cdot f_{HFCLK}/f_{PERCLK} + 1, \text{ best-case read accesses}$$

$$N_{bus\ cycles} = (N_{slave\ cycles} + 1) \cdot f_{HFCLK}/f_{PERCLK} - 1, \text{ worst-case write accesses}$$

$$N_{bus\ cycles} = (N_{slave\ cycles} + 1) \cdot f_{HFCLK}/f_{PERCLK}, \text{ worst-case read accesses}$$

where $N_{slave\ cycles}$ is the throughput of the slave's bus interface in number of PERCLK cycles per transfer, including any wait cycles introduced by the slave.

Figure 4.4. Bus Access Latency (General Case)

Note that a latency of **1** cycle corresponds to **0** wait states.

Additionally, for back-to-back accesses to the same peripheral, the throughput in number of cycles per transfer is given by:

$$N_{bus\ cycles} = N_{slave\ cycles} \cdot f_{HFCLK}/f_{PERCLK}, \text{ write accesses}$$

$$N_{bus\ cycles} = (N_{slave\ cycles} + 1) \cdot f_{HFCLK}/f_{PERCLK}, \text{ read accesses}$$

Figure 4.5. Bus Access Throughput (Back-to-Back Transfers)

Lastly, in the highest performing case, where PERCLK equals HFCLK and the slave does not introduce any additional wait states, the access latency in number of cycles is given by:

$$N_{bus\ cycles} = 1, \text{ write accesses}$$

$$N_{bus\ cycles} = 2, \text{ read accesses}$$

Figure 4.6. Bus Access Latency (Max Performance)

4.2.5.2.2 WS1 Mode

In general, when accessing a peripheral, the latency in number of HFCLK cycles, not including master arbitration, is given by:

$$N_{bus\ cycles} = N_{slave\ cycles} \cdot f_{HFCLK}/f_{PERCLK} + 2, \text{ best-case write accesses}$$

$$N_{bus\ cycles} = N_{slave\ cycles} \cdot f_{HFCLK}/f_{PERCLK} + 1, \text{ best-case read accesses}$$

$$N_{bus\ cycles} = (N_{slave\ cycles} + 1) \cdot f_{HFCLK}/f_{PERCLK} + 1, \text{ worst-case write accesses}$$

$$N_{bus\ cycles} = (N_{slave\ cycles} + 1) \cdot f_{HFCLK}/f_{PERCLK}, \text{ worst-case read accesses}$$

where $N_{slave\ cycles}$ is the throughput of the slave's bus interface in number of PERCLK cycles per transfer, including any wait cycles introduced by the slave.

Figure 4.7. Bus Access Latency (General Case)

Note that a latency of **1** cycle corresponds to **0** wait states.

Additionally, for back-to-back accesses to the same peripheral, the throughput in number of cycles per transfer is given by:

$$N_{bus\ cycles} = \max\{f_{HFCLK}/f_{PERCLK}, 2\} + N_{slave\ cycles} \cdot f_{HFCLK}/f_{PERCLK}, \text{ write accesses}$$

$$N_{bus\ cycles} = (N_{slave\ cycles} + 1) \cdot f_{HFCLK}/f_{PERCLK}, \text{ read accesses}$$

Figure 4.8. Bus Access Throughput (Back-to-Back Transfers)

Lastly, in the highest performing case, where PERCLK equals HFCLK and the slave does not introduce any additional wait states, the access latency in number of cycles is given by:

$$N_{bus\ cycles} = 3, \text{ write accesses}$$

$$N_{bus\ cycles} = 2, \text{ read accesses}$$

Figure 4.9. Bus Access Latency (Max Performance)

4.2.5.2.3 Core Access Latency

Note that the cycle counts in the equations above is in terms of the HFCLK. When the core is prescaled from the bus clock, the core will see a reduced number of latency cycles given by:

$$N_{core\ cycles} = \text{ceiling}(N_{bus\ cycles} \cdot f_{HFCORECLK}/f_{HFCLK})$$

where master arbitration is not included.

Figure 4.10. Core Access Latency

4.2.5.3 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 appropriately in MSC_CTRL.
- The core attempts to access a peripheral or system device that has its clock disabled. These faults can be enabled or disabled by setting the CLKDISFAULTEN bit appropriately in MSC_CTRL.
- The bus times out during an access. For example, this could happen while trying to synchronize volatile read data during an LE peripheral access. See 10.3.1.1 HFCLK - High Frequency Clock. These faults can be enabled or disabled by setting the TIMEOUTFAULTEN bit appropriately in MSC_CTRL.
- The EBI detects an invalid transaction. These faults can be enabled or disabled by setting the EBIFAULTEN bit appropriately in MSC_CTRL.
- There is a two-bit ECC error in the SRAM. These faults can be enabled or disabled by setting the RAMECCERRFAULTEN bit appropriately in MSC_CTRL.

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.

Note: The icache flush is not triggered at the event of a bus fault. As a result, when an instruction fetch results in a bus fault, invalid data may be cached. This means that the next time the instruction that caused the bus fault is fetched, the processor core will get the invalid cached data without any bus fault. In order to avoid invalid cached data propagation to the processor core, software should manually invalidate cache by writing 1 to MSC_CMD_INVCACHE bitfield at the event of a bus fault.

4.3 Access to Low Energy Peripherals (Asynchronous Registers)

The Low Energy Peripherals are capable of running when the high frequency oscillator and core system is powered off, i.e. in energy mode EM2 DeepSleep and in some cases also EM3 Stop. This enables the peripherals to perform tasks while the system energy consumption is minimal.

The Low Energy Peripherals are listed in Table 4.3 Low Energy Peripherals on page 59.

All Low Energy Peripherals are memory mapped, with automatic data synchronization. Because the Low Energy Peripherals are running on clocks asynchronous to the high frequency system clock, there are some constraints on how register accesses are performed, as described in the following sections.

4.3.1 Writing

Every Low Energy Peripheral has one or more registers with data that needs to be synchronized into the Low Energy clock domain to maintain data consistency and predictable operation. There are two different synchronization mechanisms on the EFM32GG12, immediate synchronization, and delayed synchronization. Immediate synchronization is available for the RTCC, LESENSE, RTC and LETIMER, and results in an immediate update of the target registers. Delayed synchronization is used for the remaining Low Energy Peripherals, and for these peripherals, a write operation requires 3 positive edges of the clock on the Low Energy Peripheral being accessed. Registers requiring synchronization are marked "Async Reg" in their description header.

Note: On the Gecko series of devices, all LE peripherals are subject to delayed synchronization.

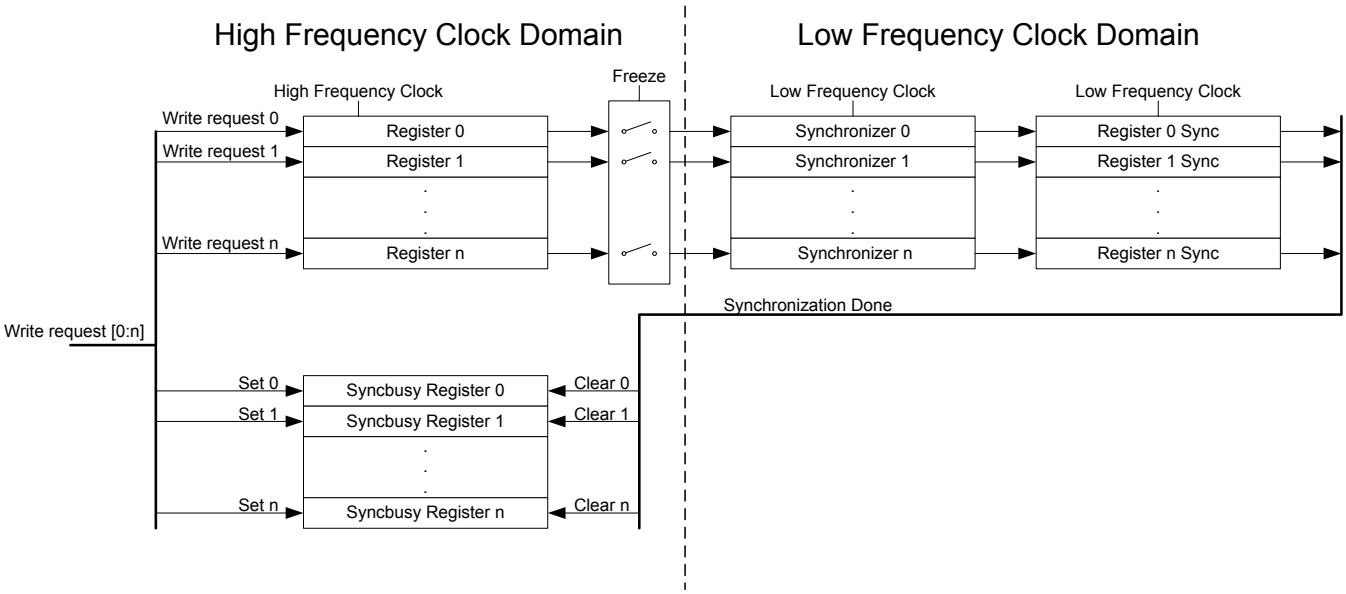


Figure 4.11. Write Operation to Low Energy Peripherals

4.3.1.1 Delayed Synchronization

After writing data to a register which value is to be synchronized into the Low Energy Peripheral using delayed synchronization, a corresponding busy flag in the <module_name>_SYNCBUSY register (e.g. LETIMER_SYNCBUSY) is set. This flag is set as long as synchronization is in progress and is cleared upon completion.

Note: Subsequent writes to the same register before the corresponding busy flag is cleared is not supported. Write before the busy flag is cleared may result in undefined behavior. In general the SYNCBUSY register only needs to be observed if there is a risk of multiple write access to a register (which must be prevented). It is not required to wait until the relevant flag in the SYNCBUSY register is cleared after writing a register. E.g., EM2 DeepSleep can be entered directly after writing a register.

See Figure 4.12 Write Operation to Low Energy Peripherals on page 64 for an overview of the writing mechanism operation.

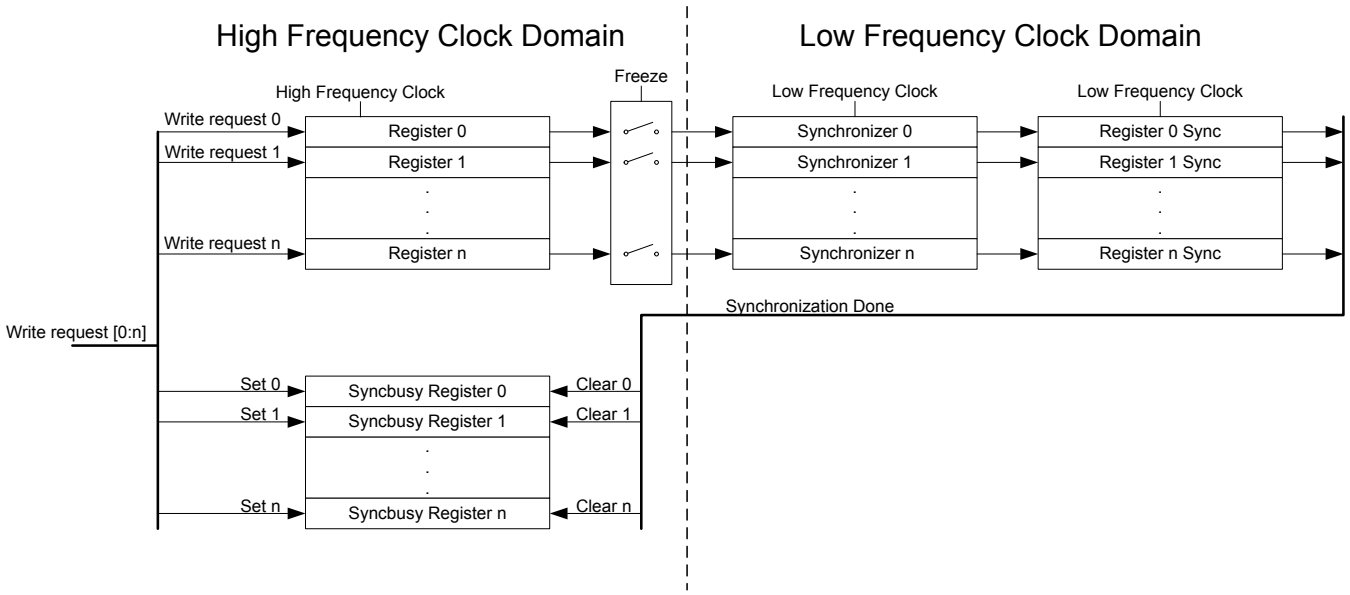


Figure 4.12. Write Operation to Low Energy Peripherals

4.3.1.2 Immediate Synchronization

In contrast to the peripherals with delayed synchronization, peripherals with immediate synchronization do not experience a register write delay for most registers. Registers are updated immediately on the peripheral write access. If such a write is done close to an edge on the clock of the peripheral, the write can be delayed until after that clock edge. This will introduce wait-states on the peripheral access.

One exception is made for commands (writing to the CMD register) in peripherals with immediate synchronization. Peripherals with immediate synchronization each have a SYNCBUSY register with a bit for the CMD register status. Commands written to a peripheral with immediate synchronization are not executed before the first peripheral clock after the write. In this period, the SYNCBUSY flag for the command register is set, indicating that the command has not yet been performed.

To maintain compatibility with earlier Gecko series, the SYNCBUSY register reserves placeholders where other register synchronization bits resided. These bits always read 0, indicating that register writes are always safe.

Note: If compatibility with earlier Gecko series is a requirement for a given application, the rules that apply to delayed synchronization with respect to SYNCBUSY should also be followed for the peripherals that support immediate synchronization.

4.3.2 Reading

When reading from a Low Energy Peripheral, the data read is synchronized regardless if it originates in the Low Energy clock domain or High Frequency clock domain. See Figure 4.13 Read Operation From Low Energy Peripherals on page 65 for an overview of the reading operation.

Note: Writing a register and then immediately reading the new value of the register may give the impression that the write operation is complete. This may not be the case. Refer to the SYNCBUSY register for correct status of the write operation to the Low Energy Peripheral.

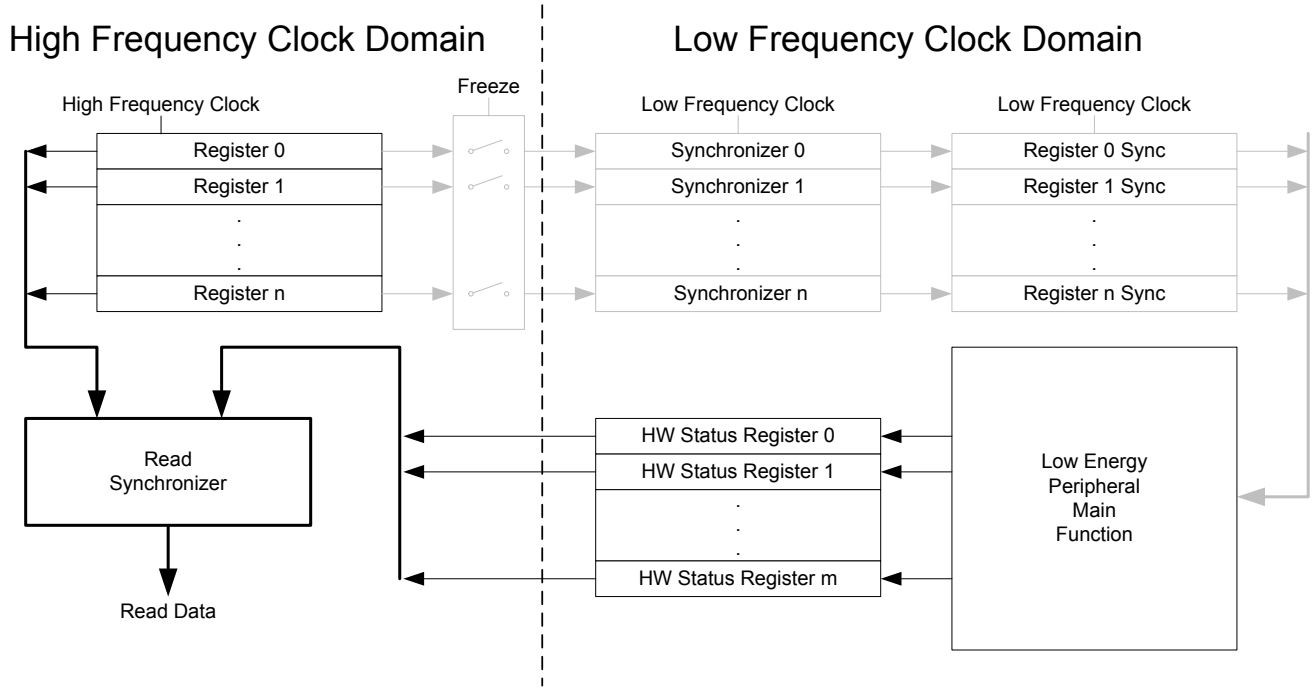


Figure 4.13. Read Operation From Low Energy Peripherals

4.3.3 FREEZE Register

In all Low Energy Peripheral with delayed synchronization there is a <module_name>_FREEZE register (e.g. RTCC_FREEZE). The register contains a bit named REGFREEZE. If precise control of the synchronization process is required, this bit may be utilized. When REGFREEZE is set, the synchronization process is halted allowing the software to write multiple Low Energy registers before starting the synchronization process, thus providing precise control of the module update process. The synchronization process is started by clearing the REGFREEZE bit.

Note: The FREEZE register is also present on peripherals with immediate synchronization, but there it has no effect

4.4 Flash

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

- Up to 1024 KB of memory
- Page size of 2 KB (minimum erase unit)
- Minimum 10K erase cycles endurance
- Application code area organized as two equal-sized banks with bank address swap capabilites.
- Greater than 10 years data retention at 85 °C
- Lock-bits for memory protection
- Data retention in any state

4.5 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.

- Up to 192 KB of memory
- Bit-band access support
- Set of RAM blocks may be powered down when not in use
- Data retention of the entire memory in EM0 Active to EM3 Stop
- ECC available on: RAM0, RAM1, RAM2

The SRAM memory may be split among two or more different AHB slaves (e.g., RAM0, RAM1, ...) in order to allow simultaneous access to different sections of the memory from two different AHB masters. For example, the Cortex-M4 can access RAM0 while the DMA controller accesses RAM1 in parallel. See 4.1 Introduction for AHB slave connectivity details.

Note: The individual RAM sections may be smaller on some parts, however, the RAM AHB slaves maintain a contiguous address map. For example, if RAM0 is half-size on a part, then RAM1 is relocated to begin immediately after RAM0's last address. Using the provided software header files and linker scripts allows handling of this remapping in an autonomous manner.

4.5.1 ECC (Error Correcting Code)

ECC provides detection of 2-bit and 1-bit SRAM errors, as well as correction of 1-bit errors. To effectively use ECC, it is important to initialize the ECC bits in the RAM. To initialize and the ECC bits, code should perform the following sequence:

1. Enable ECC writes by setting the RAMnECCEWEN bit(s) in MSC_ECCCTRL to 1.
2. Clear the RAM by writing zero to every RAM location.

Typically, this initialization sequence will be performed by a startup routine which clears the RAM prior to stack initialization.

After initialization, the ECC function can be enabled by by setting the RAMnECCCHKEN bit(s) in MSC_ECCCTRL to 1. Once enabled, the ECC hardware will detect and report 1-bit and 2-bit errors.

1-bit ECC errors are automatically corrected by the ECC hardware when it is active.

1-bit and 2-bit ECC errors are reported via the associated interrupt flags in the MSC_IF register. These flags will trigger an MSC interrupt if the corresponding enable bit in MSC_IEN is set to 1. When an ECC error is reported, software may determine the address of the error by reading MSC_RAMnECCADDR.

In addition to normal interrupt generation, a 2-bit ECC error can optionally generate a bus fault condition. 2-bit errors are not correctable, and indicate that the data is invalid. ECC bus faults are enabled by setting the RAMECCERRFAULTEN bit in MSC_CTRL to 1.

4.6 DI Page Entry Map

The DI page contains production calibration data as well as device identification information. See the peripheral chapters for how each calibration value is to be used with the associated peripheral.

The offset address is relative to the start address of the DI page (see 6.3 Functional Description).

Offset	Name	Type	Description
0x000	CAL	RO	CRC of DI-page and calibration temperature
0x004	MODULEINFO	RO	Module trace information
0x008	MODXOCAL	RO	Module Crystal Oscillator Calibration
0x020	EXTINFO	RO	External Component description
0x028	EUI48L	RO	EUI48 OUI and Unique identifier
0x02C	EUI48H	RO	OUI
0x030	CUSTOMINFO	RO	Custom information
0x034	MEMINFO	RO	Flash page size and misc. chip information
0x040	UNIQUEL	RO	Low 32 bits of device unique number
0x044	UNIQUEH	RO	High 32 bits of device unique number
0x048	MSIZE	RO	Flash and SRAM Memory size in kB
0x04C	PART	RO	Part description
0x050	DEVINFOREV	RO	Device information page revision
0x054	EMUTEMP	RO	EMU Temperature Calibration Information
0x060	ADC0CAL0	RO	ADC0 calibration register 0
0x064	ADC0CAL1	RO	ADC0 calibration register 1
0x068	ADC0CAL2	RO	ADC0 calibration register 2
0x06C	ADC0CAL3	RO	ADC0 calibration register 3
0x070	ADC1CAL0	RO	ADC1 calibration register 0
0x074	ADC1CAL1	RO	ADC1 calibration register 1
0x078	ADC1CAL2	RO	ADC1 calibration register 2
0x07C	ADC1CAL3	RO	ADC1 calibration register 3
0x080	HFRCOCAL0	RO	HFRCO Calibration Register (4 MHz)
0x08C	HFRCOCAL3	RO	HFRCO Calibration Register (7 MHz)
0x098	HFRCOCAL6	RO	HFRCO Calibration Register (13 MHz)
0x09C	HFRCOCAL7	RO	HFRCO Calibration Register (16 MHz)
0x0A0	HFRCOCAL8	RO	HFRCO Calibration Register (19 MHz)
0x0A8	HFRCOCAL10	RO	HFRCO Calibration Register (26 MHz)
0x0AC	HFRCOCAL11	RO	HFRCO Calibration Register (32 MHz)
0x0B0	HFRCOCAL12	RO	HFRCO Calibration Register (38 MHz)
0x0B4	HFRCOCAL13	RO	HFRCO Calibration Register (48 MHz)
0x0B8	HFRCOCAL14	RO	HFRCO Calibration Register (56 MHz)
0x0BC	HFRCOCAL15	RO	HFRCO Calibration Register (64 MHz)

Offset	Name	Type	Description
0x0C0	HFRCOCAL16	RO	HFRCO Calibration Register (72 MHz)
0x0E0	AUXHFRCOCAL0	RO	AUXHFRCO Calibration Register (4 MHz)
0x0EC	AUXHFRCOCAL3	RO	AUXHFRCO Calibration Register (7 MHz)
0x0F8	AUXHFRCOCAL6	RO	AUXHFRCO Calibration Register (13 MHz)
0x0FC	AUXHFRCOCAL7	RO	AUXHFRCO Calibration Register (16 MHz)
0x100	AUXHFRCOCAL8	RO	AUXHFRCO Calibration Register (19 MHz)
0x108	AUXHFRCOCAL10	RO	AUXHFRCO Calibration Register (26 MHz)
0x10C	AUXHFRCOCAL11	RO	AUXHFRCO Calibration Register (32 MHz)
0x110	AUXHFRCOCAL12	RO	AUXHFRCO Calibration Register (38 MHz)
0x114	AUXHFRCOCAL13	RO	AUXHFRCO Calibration Register (48 MHz)
0x118	AUXHFRCOCAL14	RO	AUXHFRCO Calibration Register (50 MHz)
0x140	VMONCAL0	RO	VMON Calibration Register 0
0x144	VMONCAL1	RO	VMON Calibration Register 1
0x148	VMONCAL2	RO	VMON Calibration Register 2
0x158	IDAC0CAL0	RO	IDAC0 Calibration Register 0
0x15C	IDAC0CAL1	RO	IDAC0 Calibration Register 1
0x168	DCDCLNVCTRL0	RO	DCDC Low-noise VREF Trim Register 0
0x16C	DCDCLPVCTRL0	RO	DCDC Low-power VREF Trim Register 0
0x170	DCDCLPVCTRL1	RO	DCDC Low-power VREF Trim Register 1
0x174	DCDCLPVCTRL2	RO	DCDC Low-power VREF Trim Register 2
0x178	DCDCLPVCTRL3	RO	DCDC Low-power VREF Trim Register 3
0x17C	DCDCLPCMPHYSSEL0	RO	DCDC LPCMPHYSSEL Trim Register 0
0x180	DCDCLPCMPHYSSEL1	RO	DCDC LPCMPHYSSEL Trim Register 1
0x184	VDAC0MAINCAL	RO	VDAC0 Cals for Main Path
0x188	VDAC0ALTCAL	RO	VDAC0 Cals for Alternate Path
0x18C	VDAC0CH1CAL	RO	VDAC0 CH1 Error Cal
0x190	OPA0CAL0	RO	OPA0 Calibration Register for DRIVESTRENGTH 0, INCBW=1
0x194	OPA0CAL1	RO	OPA0 Calibration Register for DRIVESTRENGTH 1, INCBW=1
0x198	OPA0CAL2	RO	OPA0 Calibration Register for DRIVESTRENGTH 2, INCBW=1
0x19C	OPA0CAL3	RO	OPA0 Calibration Register for DRIVESTRENGTH 3, INCBW=1
0x1A0	OPA0CAL4	RO	OPA0 Calibration Register for DRIVESTRENGTH 0, INCBW=0
0x1A4	OPA0CAL5	RO	OPA0 Calibration Register for DRIVESTRENGTH 1, INCBW=0
0x1A8	OPA0CAL6	RO	OPA0 Calibration Register for DRIVESTRENGTH 2, INCBW=0
0x1AC	OPA0CAL7	RO	OPA0 Calibration Register for DRIVESTRENGTH 3, INCBW=0
0x1B0	OPA1CAL0	RO	OPA1 Calibration Register for DRIVESTRENGTH 0, INCBW=1
0x1B4	OPA1CAL1	RO	OPA1 Calibration Register for DRIVESTRENGTH 1, INCBW=1
0x1B8	OPA1CAL2	RO	OPA1 Calibration Register for DRIVESTRENGTH 2, INCBW=1

Offset	Name	Type	Description
0x1BC	OPA1CAL3	RO	OPA1 Calibration Register for DRIVESTRENGTH 3, INCBW=1
0x1C0	OPA1CAL4	RO	OPA1 Calibration Register for DRIVESTRENGTH 0, INCBW=0
0x1C4	OPA1CAL5	RO	OPA1 Calibration Register for DRIVESTRENGTH 1, INCBW=0
0x1C8	OPA1CAL6	RO	OPA1 Calibration Register for DRIVESTRENGTH 2, INCBW=0
0x1CC	OPA1CAL7	RO	OPA1 Calibration Register for DRIVESTRENGTH 3, INCBW=0
0x1D0	OPA2CAL0	RO	OPA2 Calibration Register for DRIVESTRENGTH 0, INCBW=1
0x1D4	OPA2CAL1	RO	OPA2 Calibration Register for DRIVESTRENGTH 1, INCBW=1
0x1D8	OPA2CAL2	RO	OPA2 Calibration Register for DRIVESTRENGTH 2, INCBW=1
0x1DC	OPA2CAL3	RO	OPA2 Calibration Register for DRIVESTRENGTH 3, INCBW=1
0x1E0	OPA2CAL4	RO	OPA2 Calibration Register for DRIVESTRENGTH 0, INCBW=0
0x1E4	OPA2CAL5	RO	OPA2 Calibration Register for DRIVESTRENGTH 1, INCBW=0
0x1E8	OPA2CAL6	RO	OPA2 Calibration Register for DRIVESTRENGTH 2, INCBW=0
0x1EC	OPA2CAL7	RO	OPA2 Calibration Register for DRIVESTRENGTH 3, INCBW=0
0x1F0	OPA3CAL0	RO	OPA3 Calibration Register for DRIVESTRENGTH 0, INCBW=1
0x1F4	OPA3CAL1	RO	OPA3 Calibration Register for DRIVESTRENGTH 1, INCBW=1
0x1F8	OPA3CAL2	RO	OPA3 Calibration Register for DRIVESTRENGTH 2, INCBW=1
0x1FC	OPA3CAL3	RO	OPA3 Calibration Register for DRIVESTRENGTH 3, INCBW=1
0x200	OPA3CAL4	RO	OPA3 Calibration Register for DRIVESTRENGTH 0, INCBW=0
0x204	OPA3CAL5	RO	OPA3 Calibration Register for DRIVESTRENGTH 1, INCBW=0
0x208	OPA3CAL6	RO	OPA3 Calibration Register for DRIVESTRENGTH 2, INCBW=0
0x20C	OPA3CAL7	RO	OPA3 Calibration Register for DRIVESTRENGTH 3, INCBW=0
0x210	CSENGAINCAL	RO	Cap Sense Gain Adjustment
0x26C	USHFRCOCAL7	RO	USHFRCO Calibration Register (16 MHz)
0x27C	USHFRCOCAL11	RO	USHFRCO Calibration Register (32 MHz)
0x284	USHFRCOCAL13	RO	USHFRCO Calibration Register (48 MHz)
0x288	USHFRCOCAL14	RO	USHFRCO Calibration Register (50 MHz)
0x2B0	CURRMON5V	RO	5V Current monitor Transconductance

4.7 DI Page Entry Description

4.7.1 CAL - CRC of DI-page and calibration temperature

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
Access									RO								RO														
Name									TEMP								CRC														

Bit	Name	Access	Description
31:24	*Reserved*	*Reserved for future use*	
23:16	TEMP	RO	Calibration temperature as an usigned int in DegC (25 = 25DegC)
15:0	CRC	RO	CRC of DI-page (CRC-16-CCITT)

4.7.2 MODULEINFO - Module trace information

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
Access	RO												RO	RO	RO	RO	RO	RO							RO			RO				
Name	RESERVED1												HFXOCALVAL	LFXOCALVAL	EXPRESS	LFXO	TYPE	MODNUMBER							ANTENNA			HWREV				

Bit	Name	Access	Description
31:20	RESERVED1	RO	Reserved for future use
19	HFXOCALVAL	RO	HFXO Calibration Valid
	Value	Mode	Description
	0	VALID	XOCAL_HFXOCTUNE is valid
	1	NOTVALID	XOCAL_HFXOCTUNE is not valid
18	LFXOCALVAL	RO	LFXO Calibration Valid
	Value	Mode	Description
	0	VALID	XOCAL_LFXOTUNING is valid
	1	NOTVALID	XOCAL_LFXOTUNING is not valid
17	EXPRESS	RO	Blue Gecko Express
	Value	Mode	Description
	0	SUPPORTED	Blue Gecko Express is supported
	1	NONE	Blue Gecko Express is not supported
16	LFXO	RO	Module has LFXO
	Value	Mode	Description
	0	NONE	LFXO is not installed
	1	PRESENT	LFXO is installed
15	TYPE	RO	Module Type
	Value	Mode	Description
	0	PCB	PCB
	1	SIP	SIP

Bit	Name	Access	Description
14:8	MODNUMBER	RO	Module Numbers, indicates radio performance and frequency band.
7:5	ANTENNA	RO	Module Antenna Type

Value	Mode	Description
0	BUILTIN	Built-in Antenna
1	CONNECTOR	RF Connector
2	RFPAD	RF Pad

4:0	HWREV	RO	Module Hardware Revision. Starting from 0.

4.7.3 MODXOCAL - Module Crystal Oscillator Calibration

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
Access	RO							RO																								
Name	LFXOTUNING							HFXOCTUNE																								

Bit	Name	Access	Description
15:9	LFXOTUNING	RO	Calibration for LFXO TUNING
8:0	HFXOCTUNE	RO	Calibration for HFXO CTUNE

4.7.4 EXTINFO - External Component description

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
Access									RO								RO								RO							
Name									REV								CONNECTION								TYPE							

Bit	Name	Access	Description
31:24	*Reserved*	*Reserved for future use*	
23:16	REV	RO	MCM Revision. Refer to MODULEINFO instead, if available
	Value	Mode	Description
	1	REV1	Revision 1
	255	NONE	No external component present
15:8	CONNECTION	RO	Connection protocol to external component
	Value	Mode	Description
	1	SPI	SPI control interface
	2	SDIO	SDIO interface
	255	NONE	None
7:0	TYPE	RO	
	External Component. Additional IC installed in module		
	Value	Mode	Description
	1	IS25LQ040B	IS25LQ040B-JWLE1 512kB Serial Flash
	2	AT25S041	AT25S041-DWFHT 512kB Serial Flash
	3	WF200	WF200 802.11bgn WiFi NCP
	255	NONE	None

4.7.5 EUI48L - EUI48 OUI and Unique identifier

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
Access	RO								RO																							
Name	OUI48L								UNIQUEID																							

Bit	Name	Access	Description
31:24	OUI48L	RO	Lower Octet of EUI48 Organizationally Unique Identifier
23:0	UNIQUEID	RO	Unique identifier

4.7.6 EUI48H - OUI

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
Access																	RO															
Name																	OUI48H															

Bit	Name	Access	Description
31:16	*Reserved*	*Reserved for future use*	
15:0	OUI48H	RO	Upper two Octets of EUI48 Organizationally Unique Identifier

4.7.7 CUSTOMINFO - Custom information

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
Access	RO																															
Name	PARTNO																															

Bit	Name	Access	Description
31:16	PARTNO	RO	Custom part identifier as unsigned integer (e.g. 903) 65535 for standard product
15:0	*Reserved*	*Reserved for future use*	

4.7.8 MEMINFO - Flash page size and misc. chip information

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
Access	RO								RO								RO								RO							
Name	FLASH_PAGE_SIZE								PINCOUNT								PKGTYPE								TEMPGRADE							

Bit	Name	Access	Description
31:24	FLASH_PAGE_SIZE	RO	Flash page size in bytes coded as 2 ^ ((MEM_INFO_FLASH_PAGE_SIZE + 10) & 0xFF). Ie. the value 0xFF = 512 bytes.
23:16	PINCOUNT	RO	Device pin count as unsigned integer (eg. 48)
15:8	PKGTYPE	RO	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	Access	Description
7:0	TEMPGRADE	RO	Temperature Grade of product as unsigned integer enumeration

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

4.7.9 UNIQUEL - Low 32 bits of device unique number

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
Access	RO																															
Name	UNIQUEL																															

Bit	Name	Access	Description
31:0	UNIQUEL	RO	Low 32 bits of device unique number

4.7.10 UNIQUEH - High 32 bits of device unique number

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
Access	RO																															
Name	UNIQUEH																															

Bit	Name	Access	Description
31:0	UNIQUEH	RO	High 32 bits of device unique number

4.7.11 MSIZE - Flash and SRAM Memory size in kB

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
Access	RO																RO															
Name	SRAM																FLASH															

Bit	Name	Access	Description
31:16	SRAM	RO	Ram size, kbyte count as unsigned integer (eg. 16)
15:0	FLASH	RO	Flash size, kbyte count as unsigned integer (eg. 128)

4.7.12 PART - Part description

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
Access	RO								RO								RO															
Name	PROD_REV								DEVICE_FAMILY								DEVICE_NUMBER															

Bit	Name	Access	Description
31:24	PROD_REV	RO	Production revision as unsigned integer
23:16	DEVICE_FAMILY	RO	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 1 Device Config 3
41	EFR32MG13B	EFR32 Mighty Gecko Family Series 1 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
44	EFR32BG13B	EFR32 Blue Gecko Family Series 1 Device Config 3

Bit	Name	Access	Description
	45	EFR32BG13V	EFR32 Blue Gecko Family Series 1 Device Config 3
	46	EFR32ZG13P	EFR32 Zen 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
	58	EFR32ZG14P	EFR32 Zen 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
	71	G	EFM32 Gecko Device Family
	72	EFM32GG	EFM32 Giant Gecko Device Family
	72	GG	EFM32 Giant Gecko Device Family
	73	TG	EFM32 Tiny Gecko Device Family
	73	EFM32TG	EFM32 Tiny Gecko Device Family
	74	EFM32LG	EFM32 Leopard Gecko Device Family
	74	LG	EFM32 Leopard Gecko Device Family
	75	EFM32WG	EFM32 Wonder Gecko Device Family
	75	WG	EFM32 Wonder Gecko Device Family
	76	ZG	EFM32 Zero Gecko Device Family
	76	EFM32ZG	EFM32 Zero Gecko Device Family
	77	HG	EFM32 Happy Gecko Device Family
	77	EFM32HG	EFM32 Happy Gecko Device Family
	81	EFM32PG1B	EFM32 Pearl Gecko Family Series 1 Device Config 1
	83	EFM32JG1B	EFM32 Jade Gecko Family Series 1 Device Config 1
	85	EFM32PG12B	EFM32 Pearl Gecko Family Series 1 Device Config 2
	87	EFM32JG12B	EFM32 Jade Gecko Family Series 1 Device Config 2
	100	EFM32GG11B	EFM32 Giant Gecko Family Series 1 Device Config 1
	103	EFM32TG11B	EFM32 Tiny Gecko Family Series 1 Device Config 1
	106	EFM32GG12B	EFM32 Giant Gecko Family Series 1 Device Config 2
	120	EZR32LG	EZR32 Leopard Gecko Device Family

Bit	Name	Access	Description
	121	EZR32WG	EZR32 Wonder Gecko Device Family
	122	EZR32HG	EZR32 Happy Gecko Device Family
15:0	DEVICE_NUMBER	RO	Part number as unsigned integer (e.g. 233 for EFR32BG1P**233**F256GM48-B0)

4.7.13 DEVINFOREV - Device information page revision

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
Access																									RO			RO				
Name																									MAJOR			MINOR				

Bit	Name	Access	Description
31:8	*Reserved*	*Reserved for future use*	
7:5	MAJOR	RO	Major DEVINFO revision as unsigned integer (initially 1)
4:0	MINOR	RO	Minor DEVINFO layout revision as unsigned integer (initially 0)

4.7.14 EMUTEMP - EMU Temperature 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
Access																									RO							
Name																									EMUTEMPROOM							

Bit	Name	Access	Description
31:8	*Reserved*	*Reserved for future use*	
7:0	EMUTEMPROOM	RO	EMU_TEMP temperature reading at room

4.7.15 ADC0CAL0 - ADC0 calibration 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
Access		RO							RO				RO					RO							RO				RO			
Name		GAIN2V5							NEGSEOFFSET2V5				OFFSET2V5					GAIN1V25							NEGSEOFFSET1V25				OFFSET1V25			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:24	GAIN2V5	RO	Gain for 2.5V reference
23:20	NEGSEOFFSET2V5	RO	Negative single ended offset for 2.5V reference
19:16	OFFSET2V5	RO	Offset for 2.5V reference
15	*Reserved*	*Reserved for future use*	
14:8	GAIN1V25	RO	Gain for 1.25V reference
7:4	NEGSEOFFSET1V25	RO	Negative single ended offset for 1.25V reference
3:0	OFFSET1V25	RO	Offset for 1.25V reference

4.7.16 ADC0CAL1 - ADC0 calibration 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
Access		RO							RO				RO					RO							RO				RO			
Name		GAIN5VDIFF							NEGSEOFFSET5VDIFF				OFFSET5VDIFF					GAINVDD							NEGSEOFFSETVDD				OFFSETVDD			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:24	GAIN5VDIFF	RO	Gain for for 5V differential reference
23:20	NEGSEOFFSET5VDIFF	RO	Negative single ended offset with for 5V differential reference
19:16	OFFSET5VDIFF	RO	Offset for 5V differential reference
15	*Reserved*	*Reserved for future use*	
14:8	GAINVDD	RO	Gain for VDD reference
7:4	NEGSEOFFSETVDD	RO	Negative single ended offset for VDD reference
3:0	OFFSETVDD	RO	Offset for VDD reference

4.7.17 ADC0CAL2 - ADC0 calibration register 2

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
Access																									RO				RO			
Name																									NEGSEOFFSET2XVDD				OFFSET2XVDD			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:24	*Reserved*	*Reserved for future use*	
23:20	*Reserved*	*Reserved for future use*	
19:16	*Reserved*	*Reserved for future use*	
15:8	*Reserved*	*Reserved for future use*	
7:4	NEGSEOFFSET2XVDD	RO	Negative single ended offset for 2XVDD reference
3:0	OFFSET2XVDD	RO	Offset for 2XVDD reference

4.7.18 ADC0CAL3 - ADC0 calibration register 3

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
Access																						RO										
Name																						TEMPREAD1V25										

Bit	Name	Access	Description
31:16	*Reserved*	*Reserved for future use*	
15:4	TEMPREAD1V25	RO	Temperature reading at 1V25 reference
3:0	*Reserved*	*Reserved for future use*	

4.7.19 ADC1CAL0 - ADC1 calibration register 0

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
Access		RO							RO				RO					RO							RO				RO			
Name		GAIN2V5							NEGSEOFFSET2V5				OFFSET2V5					GAIN1V25							NEGSEOFFSET1V25				OFFSET1V25			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:24	GAIN2V5	RO	Gain for 2.5V reference
23:20	NEGSEOFFSET2V5	RO	Negative single ended offset for 2.5V reference
19:16	OFFSET2V5	RO	Offset for 2.5V reference
15	*Reserved*	*Reserved for future use*	
14:8	GAIN1V25	RO	Gain for 1.25V reference
7:4	NEGSEOFFSET1V25	RO	Negative single ended offset for 1.25V reference
3:0	OFFSET1V25	RO	Offset for 1.25V reference

4.7.20 ADC1CAL1 - ADC1 calibration register 1

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
Access		RO							RO				RO					RO							RO				RO			
Name		GAIN5VDIFF							NEGSEOFFSET5VDIFF				OFFSET5VDIFF					GAINVDD							NEGSEOFFSETVDD				OFFSETVDD			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:24	GAIN5VDIFF	RO	Gain for for 5V differential reference
23:20	NEGSEOFFSET5VDIFF	RO	Negative single ended offset with for 5V differential reference
19:16	OFFSET5VDIFF	RO	Offset for 5V differential reference
15	*Reserved*	*Reserved for future use*	
14:8	GAINVDD	RO	Gain for VDD reference
7:4	NEGSEOFFSETVDD	RO	Negative single ended offset for VDD reference
3:0	OFFSETVDD	RO	Offset for VDD reference

4.7.21 ADC1CAL2 - ADC1 calibration register 2

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
Access																									RO				RO			
Name																									NEGSEOFFSET2XVDD				OFFSET2XVDD			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:24	*Reserved*	*Reserved for future use*	
23:20	*Reserved*	*Reserved for future use*	
19:16	*Reserved*	*Reserved for future use*	
15:8	*Reserved*	*Reserved for future use*	
7:4	NEGSEOFFSET2XVDD	RO	Negative single ended offset for 2XVDD reference
3:0	OFFSET2XVDD	RO	Offset for 2XVDD reference

4.7.22 ADC1CAL3 - ADC1 calibration register 3

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
Access																	RO															
Name																	TEMPREAD1V25															

Bit	Name	Access	Description
31:16	*Reserved*	*Reserved for future use*	
15:4	TEMPREAD1V25	RO	Temperature reading at 1V25 reference
3:0	*Reserved*	*Reserved for future use*	

4.7.23 HFRCOCAL0 - HFRCO Calibration Register (4 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.24 HFRCOCAL3 - HFRCO Calibration Register (7 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.25 HFRCOCAL6 - HFRCO Calibration Register (13 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.26 HFRCOCAL7 - HFRCO Calibration Register (16 MHz)

Offset	Bit Position																													
0x09C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.27 HFRCOCAL8 - HFRCO Calibration Register (19 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.28 HFRCOCAL10 - HFRCO Calibration Register (26 MHz)

Offset	Bit Position																													
0x0A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.29 HFRCOCAL11 - HFRCO Calibration Register (32 MHz)

Offset	Bit Position																													
0x0AC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.30 HFRCOCAL12 - HFRCO Calibration Register (38 MHz)

Offset	Bit Position																													
0x0B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.31 HFRCOCAL13 - HFRCO Calibration Register (48 MHz)

Offset	Bit Position																													
0x0B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.32 HFRCOCAL14 - HFRCO Calibration Register (56 MHz)

Offset	Bit Position																													
0x0B8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.33 HFRCOCAL15 - HFRCO Calibration Register (64 MHz)

Offset	Bit Position																													
0x0BC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.34 HFRCOCAL16 - HFRCO Calibration Register (72 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	HFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	HFRCO enable reference for fine tuning
26:25	CLKDIV	RO	HFRCO Clock Output Divide
24	LDOHP	RO	HFRCO LDO High Power Mode
23:21	CMPBIAS	RO	HFRCO Comparator Bias Current
20:16	FREQRANGE	RO	HFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	HFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	HFRCO Tuning Value

4.7.35 AUXHFRCOCAL0 - AUXHFRCO Calibration Register (4 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.36 AUXHFRCOCAL3 - AUXHFRCO Calibration Register (7 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.37 AUXHFRCOCAL6 - AUXHFRCO Calibration Register (13 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.38 AUXHFRCOCAL7 - AUXHFRCO Calibration Register (16 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.39 AUXHFRCOCAL8 - AUXHFRCO Calibration Register (19 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.40 AUXHFRCOCAL10 - AUXHFRCO Calibration Register (26 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.41 AUXHFRCOCAL11 - AUXHFRCO Calibration Register (32 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.42 AUXHFRCOCAL12 - AUXHFRCO Calibration Register (38 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.43 AUXHFRCOCAL13 - AUXHFRCO Calibration Register (48 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.44 AUXHFRCOCAL14 - AUXHFRCO Calibration Register (50 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	AUXHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	AUXHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	AUXHFRCO Clock Output Divide
24	LDOHP	RO	AUXHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	AUXHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	AUXHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	AUXHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	AUXHFRCO Tuning Value

4.7.45 VMONCAL0 - VMON Calibration Register 0

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
Access	RO				RO				RO				RO				RO				RO				RO				RO			
Name	ALTAVDD2V98THRESCOARSE				ALTAVDD2V98THRESFINE				ALTAVDD1V86THRESCOARSE				ALTAVDD1V86THRESFINE				AVDD2V98THRESCOARSE				AVDD2V98THRESFINE				AVDD1V86THRESCOARSE				AVDD1V86THRESFINE			

Bit	Name	Access	Description
31:28	ALTAVDD2V98THRESCOARSE	RO	ALTAVDD 2.98 V Coarse Threshold Adjust
27:24	ALTAVDD2V98THRESFINE	RO	ALTAVDD 2.98 V Fine Threshold Adjust
23:20	ALTAVDD1V86THRESCOARSE	RO	ALTAVDD 1.86 V Coarse Threshold Adjust
19:16	ALTAVDD1V86THRESFINE	RO	ALTAVDD 1.86 V Fine Threshold Adjust
15:12	AVDD2V98THRESCOARSE	RO	AVDD 2.98 V Coarse Threshold Adjust
11:8	AVDD2V98THRESFINE	RO	AVDD 2.98 V Fine Threshold Adjust
7:4	AVDD1V86THRESCOARSE	RO	AVDD 1.86 V Coarse Threshold Adjust
3:0	AVDD1V86THRESFINE	RO	AVDD 1.86 V Fine Threshold Adjust

4.7.46 VMONCAL1 - VMON Calibration Register 1

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
Access	RO				RO				RO				RO				RO				RO				RO				RO			
Name	IO02V98THRESCOARSE				IO02V98THRESFINE				IO01V86THRESCOARSE				IO01V86THRESFINE				DVDD2V98THRESCOARSE				DVDD2V98THRESFINE				DVDD1V86THRESCOARSE				DVDD1V86THRESFINE			

Bit	Name	Access	Description
31:28	IO02V98THRESCOARSE	RO	IO0 2.98 V Coarse Threshold Adjust
27:24	IO02V98THRESFINE	RO	IO0 2.98 V Fine Threshold Adjust
23:20	IO01V86THRESCOARSE	RO	IO0 1.86 V Coarse Threshold Adjust
19:16	IO01V86THRESFINE	RO	IO0 1.86 V Fine Threshold Adjust
15:12	DVDD2V98THRESCOARSE	RO	DVDD 2.98 V Coarse Threshold Adjust
11:8	DVDD2V98THRESFINE	RO	DVDD 2.98 V Fine Threshold Adjust
7:4	DVDD1V86THRESCOARSE	RO	DVDD 1.86 V Coarse Threshold Adjust
3:0	DVDD1V86THRESFINE	RO	DVDD 1.86 V Fine Threshold Adjust

4.7.47 VMONCAL2 - VMON Calibration Register 2

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
Access	RO				RO				RO				RO				RO				RO				RO				RO			
Name	IO12V98THRESCOARSE				IO12V98THRESFINE				IO11V86THRESCOARSE				IO11V86THRESFINE				BUVDD2V98THRESCOARSE				BUVDD2V98THRESFINE				BUVDD1V86THRESCOARSE				BUVDD1V86THRESFINE			

Bit	Name	Access	Description
31:28	IO12V98THRESCOARSE	RO	IO1 2.98 V Coarse Threshold Adjust
27:24	IO12V98THRESFINE	RO	IO1 2.98 V Fine Threshold Adjust
23:20	IO11V86THRESCOARSE	RO	IO1 1.86 V Coarse Threshold Adjust
19:16	IO11V86THRESFINE	RO	IO1 1.86 V Fine Threshold Adjust
15:12	BUVDD2V98THRESCOARSE	RO	BUVDD 2.98 V Coarse Threshold Adjust
11:8	BUVDD2V98THRESFINE	RO	BUVDD 2.98 V Fine Threshold Adjust
7:4	BUVDD1V86THRESCOARSE	RO	BUVDD 1.86 V Coarse Threshold Adjust
3:0	BUVDD1V86THRESFINE	RO	BUVDD 1.86 V Fine Threshold Adjust

4.7.48 IDAC0CAL0 - IDAC0 Calibration Register 0

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
Access	RO								RO								RO								RO							
Name	SOURCERANGE3TUNING								SOURCERANGE2TUNING								SOURCERANGE1TUNING								SOURCERANGE0TUNING							

Bit	Name	Access	Description
31:24	SOURCERANGE3TUNING	RO	Calibrated middle step (16) of current source mode range 3
23:16	SOURCERANGE2TUNING	RO	Calibrated middle step (16) of current source mode range 2
15:8	SOURCERANGE1TUNING	RO	Calibrated middle step (16) of current source mode range 1
7:0	SOURCERANGE0TUNING	RO	Calibrated middle step (16) of current source mode range 0

4.7.49 IDAC0CAL1 - IDAC0 Calibration Register 1

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
Access	RO								RO								RO								RO							
Name	SINKRANGE3TUNING								SINKRANGE2TUNING								SINKRANGE1TUNING								SINKRANGE0TUNING							

Bit	Name	Access	Description
31:24	SINKRANGE3TUNING	RO	Calibrated middle step (16) of current sink mode range 3
23:16	SINKRANGE2TUNING	RO	Calibrated middle step (16) of current sink mode range 2
15:8	SINKRANGE1TUNING	RO	Calibrated middle step (16) of current sink mode range 1
7:0	SINKRANGE0TUNING	RO	Calibrated middle step (16) of current sink mode range 0

4.7.50 DCDCLNVCTRL0 - DCDC Low-noise VREF Trim Register 0

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
Access	RO								RO								RO								RO							
Name	3V0LNATT1								1V8LNATT1								1V8LNATT0								1V2LNATT0							

Bit	Name	Access	Description
31:24	3V0LNATT1	RO	DCDC LNVREF Trim for 3.0V output, LNATT=1
23:16	1V8LNATT1	RO	DCDC LNVREF Trim for 1.8V output, LNATT=1
15:8	1V8LNATT0	RO	DCDC LNVREF Trim for 1.8V output, LNATT=0
7:0	1V2LNATT0	RO	DCDC LNVREF Trim for 1.2V output, LNATT=0

4.7.51 DCDCLPVCTRL0 - DCDC Low-power VREF Trim Register 0

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
Access	RO								RO								RO								RO							
Name	1V8LPATT0LPCMPBIAS1								1V2LPATT0LPCMPBIAS1								1V8LPATT0LPCMPBIAS0								1V2LPATT0LPCMPBIAS0							

Bit	Name	Access	Description
31:24	1V8LPATT0LPCMPBIAS1	RO	DCDC LPVREF Trim for 1.8V output, LPATT=0, LPCMPBIAS=1
23:16	1V2LPATT0LPCMPBIAS1	RO	DCDC LPVREF Trim for 1.2V output, LPATT=0, LPCMPBIAS=1
15:8	1V8LPATT0LPCMPBIAS0	RO	DCDC LPVREF Trim for 1.8V output, LPATT=0, LPCMPBIAS=0
7:0	1V2LPATT0LPCMPBIAS0	RO	DCDC LPVREF Trim for 1.2V output, LPATT=0, LPCMPBIAS=0

4.7.52 DCDCLPVCTRL1 - DCDC Low-power VREF Trim Register 1

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
Access	RO								RO								RO								RO							
Name	1V8LPATT0LPCMPBIAS3								1V2LPATT0LPCMPBIAS3								1V8LPATT0LPCMPBIAS2								1V2LPATT0LPCMPBIAS2							

Bit	Name	Access	Description
31:24	1V8LPATT0LPCMPBIAS3	RO	DCDC LPVREF Trim for 1.8V output, LPATT=0, LPCMPBIAS=3
23:16	1V2LPATT0LPCMPBIAS3	RO	DCDC LPVREF Trim for 1.2V output, LPATT=0, LPCMPBIAS=3
15:8	1V8LPATT0LPCMPBIAS2	RO	DCDC LPVREF Trim for 1.8V output, LPATT=0, LPCMPBIAS=2
7:0	1V2LPATT0LPCMPBIAS2	RO	DCDC LPVREF Trim for 1.2V output, LPATT=0, LPCMPBIAS=2

4.7.53 DCDCLPVCTRL2 - DCDC Low-power VREF Trim Register 2

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
Access	RO								RO								RO								RO							
Name	3V0LPATT1LPCMPBIAS1								1V8LPATT1LPCMPBIAS1								3V0LPATT1LPCMPBIAS0								1V8LPATT1LPCMPBIAS0							

Bit	Name	Access	Description
31:24	3V0LPATT1LPCMPBIAS1	RO	DCDC LPVREF Trim for 3.0V output, LPATT=1, LPCMPBIAS=1
23:16	1V8LPATT1LPCMPBIAS1	RO	DCDC LPVREF Trim for 1.8V output, LPATT=1, LPCMPBIAS=1
15:8	3V0LPATT1LPCMPBIAS0	RO	DCDC LPVREF Trim for 3.0V output, LPATT=1, LPCMPBIAS=0
7:0	1V8LPATT1LPCMPBIAS0	RO	DCDC LPVREF Trim for 1.8V output, LPATT=1, LPCMPBIAS=0

4.7.54 DCDCLPVCTRL3 - DCDC Low-power VREF Trim Register 3

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
Access	RO								RO								RO								RO							
Name	3V0LPATT1LPCMPBIAS3								1V8LPATT1LPCMPBIAS3								3V0LPATT1LPCMPBIAS2								1V8LPATT1LPCMPBIAS2							

Bit	Name	Access	Description
31:24	3V0LPATT1LPCMPBIAS3	RO	DCDC LPVREF Trim for 3.0V output, LPATT=1, LPCMPBIAS=3
23:16	1V8LPATT1LPCMPBIAS3	RO	DCDC LPVREF Trim for 1.8V output, LPATT=1, LPCMPBIAS=3
15:8	3V0LPATT1LPCMPBIAS2	RO	DCDC LPVREF Trim for 3.0V output, LPATT=1, LPCMPBIAS=3
7:0	1V8LPATT1LPCMPBIAS2	RO	DCDC LPVREF Trim for 1.8V output, LPATT=1, LPCMPBIAS=2

4.7.55 DCDCLPCMPHYSSEL0 - DCDC LPCMPHYSSEL Trim Register 0

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
Access																	RO								RO							
Name																	LPCMPHYSSELLPATT1								LPCMPHYSSELLPATT0							

Bit	Name	Access	Description
31:16	*Reserved*	*Reserved for future use*	
15:8	LPCMPHYSSELLPATT1	RO	DCDC LPCMPHYSSEL Trim, LPATT=1
7:0	LPCMPHYSSELLPATT0	RO	DCDC LPCMPHYSSEL Trim, LPATT=0

4.7.56 DCDCLPCMPHYSSEL1 - DCDC LPCMPHYSSEL Trim Register 1

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
Access	RO								RO								RO								RO							
Name	LPCMPHYSSELLPCMPBIAS3								LPCMPHYSSELLPCMPBIAS2								LPCMPHYSSELLPCMPBIAS1								LPCMPHYSSELLPCMPBIAS0							

Bit	Name	Access	Description
31:24	LPCMPHYSSELLPCMPBIAS3	RO	DCDC LPCMPHYSSEL Trim, LPCMPBIAS=3
23:16	LPCMPHYSSELLPCMPBIAS2	RO	DCDC LPCMPHYSSEL Trim, LPCMPBIAS=2
15:8	LPCMPHYSSELLPCMPBIAS1	RO	DCDC LPCMPHYSSEL Trim, LPCMPBIAS=1
7:0	LPCMPHYSSELLPCMPBIAS0	RO	DCDC LPCMPHYSSEL Trim, LPCMPBIAS=0

4.7.57 VDAC0MAINCAL - VDAC0 Cals for Main Path

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
Access			RO						RO						RO						RO						RO					
Name			GAINERRTRIMVDDANAEXTPIN						GAINERRTRIM2V5						GAINERRTRIM1V25						GAINERRTRIM2V5LN						GAINERRTRIM1V25LN					

Bit	Name	Access	Description
31:30	*Reserved*	*Reserved for future use*	
29:24	GAINERRTRIMVDDANAEXTPIN	RO	Gain Error Trim Value for DAC main output using references VDDANA and EXTPIN
23:18	GAINERRTRIM2V5	RO	Gain Error Trim Value for DAC main output using reference 2V5
17:12	GAINERRTRIM1V25	RO	Gain Error Trim Value for DAC main output using reference 1V25
11:6	GAINERRTRIM2V5LN	RO	Gain Error Trim Value for DAC main output using reference 2V5LN
5:0	GAINERRTRIM1V25LN	RO	Gain Error Trim Value for DAC main output using reference 1V25LN

4.7.58 VDAC0ALTCAL - VDAC0 Cals for Alternate Path

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
Access			RO						RO						RO						RO						RO					
Name			GAINERRTRIMVDDANAEXTPINALT						GAINERRTRIM2V5ALT						GAINERRTRIM1V25ALT						GAINERRTRIM2V5LNALT						GAINERRTRIM1V25LNALT					

Bit	Name	Access	Description
31:30	*Reserved*	*Reserved for future use*	
29:24	GAINERRTRIMVDDANAEXTPI-NALT	RO	Gain Error Trim Value for DAC alternative output using references VDDANA and EXTPIN
23:18	GAINERRTRIM2V5ALT	RO	Gain Error Trim Value for DAC alternative output using reference 2V5
17:12	GAINERRTRIM1V25ALT	RO	Gain Error Trim Value for DAC alternative output using reference 1V25
11:6	GAINERRTRIM2V5LNALT	RO	Gain Error Trim Value for DAC alternative output using reference 2V5LN
5:0	GAINERRTRIM1V25LNALT	RO	Gain Error Trim Value for DAC alternative output using reference 1V25LN

4.7.59 VDAC0CH1CAL - VDAC0 CH1 Error Cal

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
Access																					RO				RO					RO		
Name																					GAINERRTRIMCH1B				GAINERRTRIMCH1A					OFFSETTRIM		

Bit	Name	Access	Description
31:12	*Reserved*	*Reserved for future use*	
11:8	GAINERRTRIMCH1B	RO	Gain Error Trim Value for Channel 1 Main Output for references 2V5LN, 2V5
7:4	GAINERRTRIMCH1A	RO	Gain Error Trim Value for Channel 1 Main Output for references 1V25LN, 1V25, VDDANA, EXTPIN
3	*Reserved*	*Reserved for future use*	
2:0	OFFSETTRIM	RO	Input Buffer Offset Calibration Value for all DAC references

4.7.60 OPA0CAL0 - OPA0 Calibration Register for DRIVESTRENGTH 0, INCBW=1

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
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.61 OPA0CAL1 - OPA0 Calibration Register for DRIVESTRENGTH 1, INCBW=1

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
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.62 OPA0CAL2 - OPA0 Calibration Register for DRIVESTRENGTH 2, INCBW=1

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
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.63 OPA0CAL3 - OPA0 Calibration Register for DRIVESTRENGTH 3, INCBW=1

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
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.64 OPA0CAL4 - OPA0 Calibration Register for DRIVESTRENGTH 0, INCBW=0

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
Access		RO						RO						RO			RO				RO			RO						RO			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.65 OPA0CAL5 - OPA0 Calibration Register for DRIVESTRENGTH 1, INCBW=0

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
Access		RO						RO						RO			RO				RO			RO						RO			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.66 OPA0CAL6 - OPA0 Calibration Register for DRIVESTRENGTH 2, INCBW=0

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
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.67 OPA0CAL7 - OPA0 Calibration Register for DRIVESTRENGTH 3, INCBW=0

Offset	Bit Position																														
0x1AC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.68 OPA1CAL0 - OPA1 Calibration Register for DRIVESTRENGTH 0, INCBW=1

Offset	Bit Position																														
0x1B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.69 OPA1CAL1 - OPA1 Calibration Register for DRIVESTRENGTH 1, INCBW=1

Offset	Bit Position																														
0x1B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.70 OPA1CAL2 - OPA1 Calibration Register for DRIVESTRENGTH 2, INCBW=1

Offset	Bit Position																														
0x1B8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.71 OPA1CAL3 - OPA1 Calibration Register for DRIVESTRENGTH 3, INCBW=1

Offset	Bit Position																														
0x1BC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.72 OPA1CAL4 - OPA1 Calibration Register for DRIVESTRENGTH 0, INCBW=0

Offset	Bit Position																															
0x1C0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.73 OPA1CAL5 - OPA1 Calibration Register for DRIVESTRENGTH 1, INCBW=0

Offset	Bit Position																															
0x1C4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.74 OPA1CAL6 - OPA1 Calibration Register for DRIVESTRENGTH 2, INCBW=0

Offset	Bit Position																															
0x1C8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.75 OPA1CAL7 - OPA1 Calibration Register for DRIVESTRENGTH 3, INCBW=0

Offset	Bit Position																														
0x1CC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.76 OPA2CAL0 - OPA2 Calibration Register for DRIVESTRENGTH 0, INCBW=1

Offset	Bit Position																															
0x1D0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.77 OPA2CAL1 - OPA2 Calibration Register for DRIVESTRENGTH 1, INCBW=1

Offset	Bit Position																															
0x1D4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.78 OPA2CAL2 - OPA2 Calibration Register for DRIVESTRENGTH 2, INCBW=1

Offset	Bit Position																														
0x1D8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.79 OPA2CAL3 - OPA2 Calibration Register for DRIVESTRENGTH 3, INCBW=1

Offset	Bit Position																															
0x1DC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.80 OPA2CAL4 - OPA2 Calibration Register for DRIVESTRENGTH 0, INCBW=0

Offset	Bit Position																														
0x1E0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.81 OPA2CAL5 - OPA2 Calibration Register for DRIVESTRENGTH 1, INCBW=0

Offset	Bit Position																														
0x1E4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.82 OPA2CAL6 - OPA2 Calibration Register for DRIVESTRENGTH 2, INCBW=0

Offset	Bit Position																														
0x1E8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.83 OPA2CAL7 - OPA2 Calibration Register for DRIVESTRENGTH 3, INCBW=0

Offset	Bit Position																														
0x1EC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.84 OPA3CAL0 - OPA3 Calibration Register for DRIVESTRENGTH 0, INCBW=1

Offset	Bit Position																															
0x1F0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.85 OPA3CAL1 - OPA3 Calibration Register for DRIVESTRENGTH 1, INCBW=1

Offset	Bit Position																														
0x1F4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.86 OPA3CAL2 - OPA3 Calibration Register for DRIVESTRENGTH 2, INCBW=1

Offset	Bit Position																														
0x1F8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.87 OPA3CAL3 - OPA3 Calibration Register for DRIVESTRENGTH 3, INCBW=1

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
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.88 OPA3CAL4 - OPA3 Calibration Register for DRIVESTRENGTH 0, INCBW=0

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
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.89 OPA3CAL5 - OPA3 Calibration Register for DRIVESTRENGTH 1, INCBW=0

Offset	Bit Position																														
0x204	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.90 OPA3CAL6 - OPA3 Calibration Register for DRIVESTRENGTH 2, INCBW=0

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
Access		RO						RO						RO			RO				RO			RO						RO		
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2						CM1		

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.91 OPA3CAL7 - OPA3 Calibration Register for DRIVESTRENGTH 3, INCBW=0

Offset	Bit Position																														
0x20C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access		RO						RO						RO			RO				RO			RO					RO				
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Access	Description
31	*Reserved*	*Reserved for future use*	
30:26	OFFSETN	RO	OPA Inverting Input Offset Configuration Value.
25	*Reserved*	*Reserved for future use*	
24:20	OFFSETP	RO	OPA Non-Inverting Input Offset Configuration Value.
19	*Reserved*	*Reserved for future use*	
18:17	GM3	RO	Gm3 Trim Value
16	*Reserved*	*Reserved for future use*	
15:13	GM	RO	Gm Trim Value
12	*Reserved*	*Reserved for future use*	
11:10	CM3	RO	Compensation cap Cm3 trim value
9	*Reserved*	*Reserved for future use*	
8:5	CM2	RO	Compensation cap Cm2 trim value
4	*Reserved*	*Reserved for future use*	
3:0	CM1	RO	Compensation cap Cm1 trim value

4.7.92 CSENGAINCAL - Cap Sense Gain Adjustment

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
Access																									RO							
Name																									GAINCAL							

Bit	Name	Access	Description
31:8	*Reserved*	*Reserved for future use*	
7:0	GAINCAL	RO	Gain Adjustment for Cap Sense. Gain should be scaled by GAINCAL/128

4.7.93 USHFRCOCAL7 - USHFRCO Calibration Register (16 MHz)

Offset	Bit Position																													
0x26C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	USHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	USHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	USHFRCO Clock Output Divide
24	LDOHP	RO	USHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	USHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	USHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	USHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	USHFRCO Tuning Value

4.7.94 USHFRCOCAL11 - USHFRCO Calibration Register (32 MHz)

Offset	Bit Position																													
0x27C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	USHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	USHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	USHFRCO Clock Output Divide
24	LDOHP	RO	USHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	USHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	USHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	USHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	USHFRCO Tuning Value

4.7.95 USHFRCOCAL13 - USHFRCO Calibration Register (48 MHz)

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
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Access	Description
31:28	VREFTC	RO	USHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	USHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	USHFRCO Clock Output Divide
24	LDOHP	RO	USHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	USHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	USHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	USHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	USHFRCO Tuning Value

4.7.96 USHFRCOCAL14 - USHFRCO Calibration Register (50 MHz)

Offset	Bit Position																													
0x288	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access	RO				RO	RO		RO	RO			RO							RO							RO					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

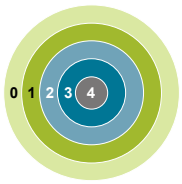
Bit	Name	Access	Description
31:28	VREFTC	RO	USHFRCO Temperature Coefficient Trim on Comparator Reference
27	FINETUNINGEN	RO	USHFRCO enable reference for fine tuning
26:25	CLKDIV	RO	USHFRCO Clock Output Divide
24	LDOHP	RO	USHFRCO LDO High Power Mode
23:21	CMPBIAS	RO	USHFRCO Comparator Bias Current
20:16	FREQRANGE	RO	USHFRCO Frequency Range
15:14	*Reserved*	*Reserved for future use*	
13:8	FINETUNING	RO	USHFRCO Fine Tuning Value
7	*Reserved*	*Reserved for future use*	
6:0	TUNING	RO	USHFRCO Tuning Value

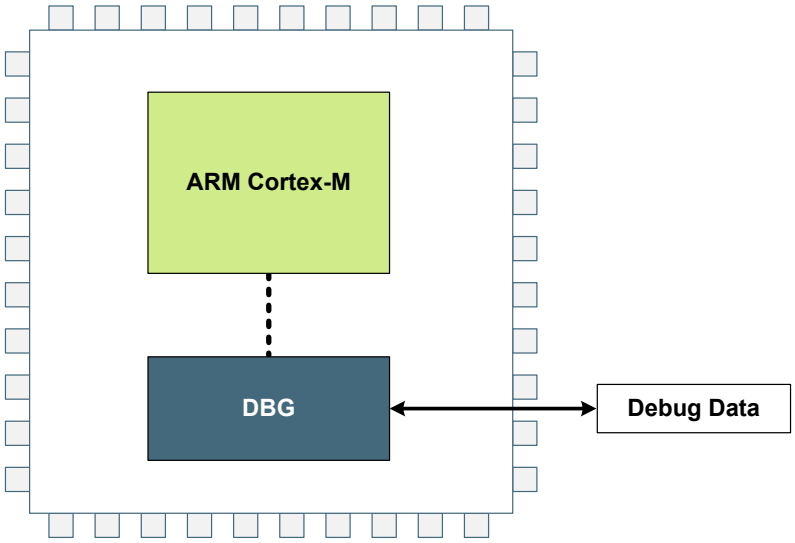
4.7.97 CURRMON5V - 5V Current monitor Transconductance

Offset	Bit Position																													
0x2B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Access																																
Name																																

Bit	Name	Access	Description
31:16	*Reserved*	*Reserved for future use*	
15:0	*Reserved*	*Reserved for future use*	

5. DBG - Debug Interface





Quick Facts

What?

The Debug Interface is used to program and debug EFM32 Giant Gecko 12 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-M4 supports advanced debugging features. EFM32 Giant Gecko 12 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.

5.1 Introduction

The EFM32 Giant Gecko 12 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.

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

- ARM Cortex-M4 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

5.2 Features

- Debug Access Port Serial Wire JTAG (DAP SWJ-DP)
  - Implements the ADIv5 debug interface
- Authentication Access Point (AAP)
  - Implements various user commands
- Flash Patch and Breakpoint (FPB) unit
  - Implement breakpoints and code patches
- Data Watch point and Trace (DWT) unit
  - Implement 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

5.3 Functional Description

Operation of the available debug interface is described in the following sections.

5.3.1 Debug Pins

The following pins are the debug connections for the device:

- Serial Wire Clock Input and Test Clock Input (SWCLKTCK) : 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) : 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.

The debug pins have pull-down and pull-up enabled by default, so leaving them enabled may increase the current consumption if left connected to supply or ground. The debug pins can be enabled and disabled through GPIO_ROUTEPEN, see 34.3.4.2.3 Disabling Debug Connections. Remember that upon disabling the debug pins, debug contact with the device is lost once the DAP SWJ-DP power request bits are deasserted. By default after power cycle the part's debug pins are in JTAG mode. If during debugging session the pins are switched to SWD mode, a power cycle is required to bring restore the pins to JTAG mode.

5.3.2 Embedded Trace Macrocell V3.5 (ETM)

The ETM makes it possible to trace both instruction and data from the processor in real time. The 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. The ETM circuit does not automatically enable an auxiliary clock. If AUXCLK is elected as the ETM clock source, the AUXHFRCO oscillator must be enabled by setting AUXHFRCOEN in CMU_OSCENCMD before enabling the ETM. Likewise, if switching from AUXCLK to a different source for ETM, the AUXHFRCO must remain enabled until the clock switch has completed.

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 - Trace Data 3 (TD0-TD3): The data pins provide the compressed trace stream. These pins are 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 can be enabled through the GPIO. For more information on how to enable the ETM Trace pins, the reader is referred to 34.3.4.2.4 ETM Trace Connections.

5.3.3 Debug and EM2 DeepSleep/EM3 Stop

Leaving the debugger connected when issuing a WFI or WFE to enter EM2 DeepSleep or EM3 Stop will make the system enter a special EM2 DeepSleep. This mode differs from regular EM2 DeepSleep and EM3 Stop 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 Sleep and it is therefore important to deassert the power requests in the DAP SWJ-DP and disconnect the debugger before doing current consumption measurements.

5.3.4 Authentication Access Point

The Authentication Acces Point (AAP) is a set of registers that provide a minimal amount of debugging and system level commands. The AAP registers contain commands to issue a FLASH erase, a system reset, a CRC of user code pages, and stalling the system bus. The user must program the APSEL bit field to 255 inside of the ARM DAP SWJ Debug Port SELECT register to access the AAP. The AAP is only accessible from a debugger and not from the core.

5.3.4.1 System Bus Stall

The system bus can be stalled at any time using the SYSBUSSTALL register bit. Once the SYSBUSSTALL is set, the system bus will remain stalled until SYSBUSSTALL is cleared. While the system bus is stalled, only the registers inside the Cortex-M4, AAP and the debugger can be accessed. The SYSBUSSTALL register is available at all times through the AAP.

5.3.4.2 Command Key

The AAP uses a command key to enable the DEVICEERASE and SYSRESETREQ AAP commands. The command key must be written with the correct key in order for the commands to execute.

5.3.4.3 Device Erase

The device can be erased by stalling the system bus, writing AAP_CMDKEY, and then writing the DEVICEERASE register bit. Upon writing the command bit, the ERASEBUSY bit is asserted. The ERASEBUSY bit will be de-asserted once the erase is complete. The SYSRESETREQ bit must then be set to resume a normal debugger session. The DEVICEERASE register is available at all times through the AAP once the CMDKEY is entered.

5.3.4.4 System Reset

The system can be reset by writing AAP_CMDKEY followed by writing the SYSRESTREQ register bit. This must be done after asserting DEVICEERASE or CRCREQ. Depending on the reset level setting for system reset, asserting SYSRESETREQ will either reset the entire AAP register space or just the SYSRESETREQ bit. See 8.3.1 Reset Levels for more details on reset levels. The SYSRESETREQ register is available at all times through the AAP once the CMDKEY is enetered.

5.3.4.5 User Flash Page CRC

The CRCREQ command initiates a CRC calculation on a given Flash Page. The CRC is only available on the Main, User Data, and Lock Bit pages. It is highly recommended that the system bus is stalled before any CRCREQ commands are issued. The CRC calculation uses the on chip CRC block configured in 32 bit CRC mode. The Flash Page address for the CRCREQ command is written to the CRCADDR register. After issuing the CRCREQ, the CRCBUSY flag is asserted. Once the CRCBUSY flag is de-asserted, the resulting page CRC can be found in the CRCRESULT register. Once issuing a CRC command, the CPU is stalled and remains stalled until a system reset occurs. Multiple CRC requests can occur before resetting the system. However, a CRC request that occurs while the CRCBUSY flag is asserted will be ignored. The CRC registers are available at all times through the AAP.

5.3.5 Debug Lock

The debug access to the Cortex-M4 is locked by clearing the Debug Lock Word (DLW) and resetting the device, see 6.3.2 Lock Bits (LB) Page Description.

When debug access is locked, the debugger can access the DAP SWJ-DP and AAP registers. However, the connection to the Cortex-M4 core and the whole bus-system is blocked. This mechanism is controlled by the Authentication Access Port (AAP) as illustrated by Figure 5.1 AAP - Authentication Access Port on page 160.

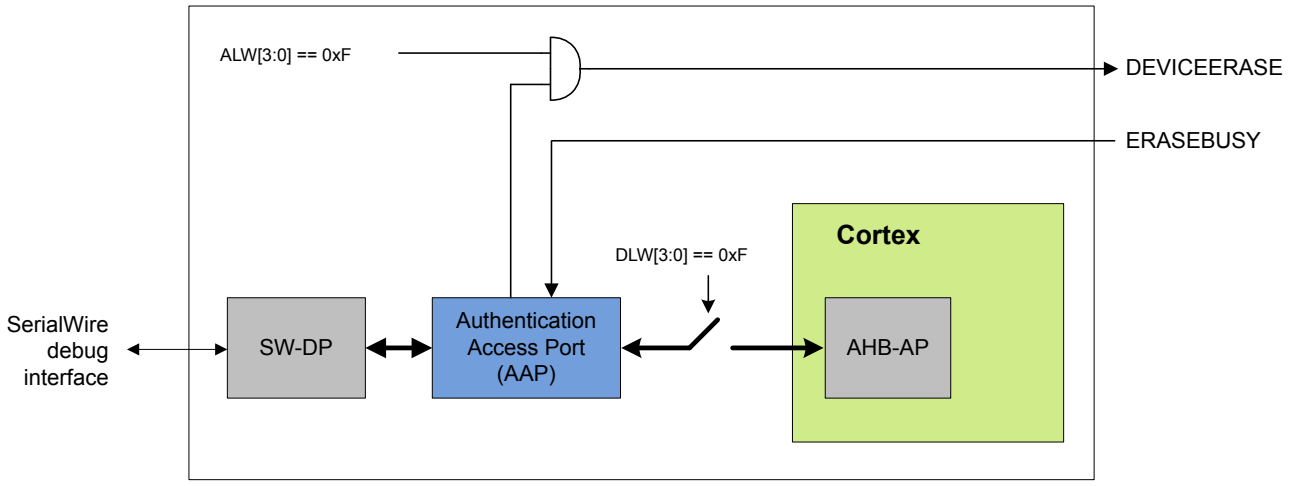


Figure 5.1. AAP - Authentication Access Port

If the DLW is cleared, the device is locked. If the device is locked and the the AAP Lock Word (ALW) has not been cleared, it can be unlocked by writing a valid key to the AAP_CMDKEY register and then setting the DEVICEERASE bit of the AAP_CMD register via the debug interface. This operation erases the main block of flash, clears all lock bits, and debug access to the Cortex-M4 and bus-system is enabled. The operation takes tens of mili seconds to complete. Note that the SRAM contents will also be deleted during a device erase, while the UD-page is not erased.

The debugger may read the status of the device erase from the AAP_STATUS register. When the ERASEBUSY bit is set low after DEVICEERASE of the AAP_CMD register is set, the debugger may set the SYSRESETREQ bit in the AAP_CMD register. After reset, the debugger may resume a normal debug session through the AHB-AP.

5.3.6 AAP Lock

Take extreme caution when using this feature. Once the AAP has been locked, the state of the FLASH can not be changed via the debugger.

5.3.7 Debugger Reads of Actionable Registers

Some peripheral registers cause particular actions when read, e.g FIFOs which pop and IFC registers which clear the IF flags when read. This can cause problems when debugging and the user wants to read the value without triggering the read action. For this reason, by default, the peripherals will not execute these triggered actions when an attached debugger is performing the read accesses through the AAP. To override this behavior, the debugger can configure the MASTERTYPE bitfield of the Cortex-M4 AHB Access Port CSW register in order to emulate a core access when performing system bus transfers.

Note:

- Registers with actionable reads are noted in their register descriptions. Refer to Table 1.1 Register Access Types on page 39.
- The following peripherals do not respect the debugger master override, and so may still cause their triggered actions to occur (e.g., when reading IFC):
  - QSPI
  - USB
  - SDIO
  - CAN

5.3.8 Debug Recovery

Debug recovery is the ability to stall the system bus before the Cortex-M4 executes code. For example, the first few instructions may disconnect the debugger pins. When this occurs it is difficult to connect the debugger and halt the Cortex-M4 before the Cortex-M4 starts to execute. By holding down pin reset, issuing the System Bus Stall AAP instruction, then releasing pin reset, the debugger can stall the system bus before the Cortex-M4 has a chance to execute. Because the system is under reset during this procedure the Debugger can not look for ACK's from the part. Once the system bus is stalled, the FLASH can be erased by issuing the AAP_CMDKEY and then the writting the DEVICEERASE in the AAP_CMD register.

5.4 Register Map

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

Offset	Name	Type	Description
0x000	AAP_CMD	W1	Command Register
0x004	AAP_CMDKEY	W1	Command Key Register
0x008	AAP_STATUS	R	Status Register
0x00C	AAP_CTRL	RW	Control Register
0x010	AAP_CRCCMD	W1	CRC Command Register
0x014	AAP_CRCSTATUS	R	CRC Status Register
0x018	AAP_CRCADDR	RW	CRC Address Register
0x01C	AAP_CRCRESULT	R	CRC Result Register
0x0FC	AAP_IDR	R	AAP Identification Register

5.5 Register Description

5.5.1 AAP_CMD - Command 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																															0	0
Access																															W1	W1
Name																															SYSRESETREQ	DEVICEERASE

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	SYSRESETREQ	0	W1	**System Reset Request** A system reset request is generated when set to 1. This register is write enabled from the AAP_CMDKEY register.
0	DEVICEERASE	0	W1	**Erase the Flash Main Block, SRAM and Lock Bits** When set, all data and program code in the main block is erased, the SRAM is cleared and then the Lock Bit (LB) page is erased. This also includes the Debug Lock Word (DLW), causing debug access to be enabled after the next reset. The information block User Data page (UD) is left unchanged, but the User data page Lock Word (ULW) is erased. This register is write enabled from the AAP_CMDKEY register.

5.5.2 AAP_CMDKEY - Command Key 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																0x00000000																
Access																W1																
Name																WRITEKEY																

Bit	Name	Reset	Access	Description
31:0	WRITEKEY	0x00000000	W1	**CMD Key Register** The key value must be written to this register to write enable the AAP_CMD register.

Value	Mode	Description
0xCFACC118	WRITEEN	Enable write to AAP_CMD

5.5.3 AAP_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																															0	0
Access																															R	R
Name																															LOCKED	ERASEBUSY

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	LOCKED	0	R	**AAP Locked**
	Set when the AAP is locked, .e.g the AAP Lock Word AAP lsb bits are not 0xF			
0	ERASEBUSY	0	R	**Device Erase Command Status**
	This bit is set when a device erase is executing.			

5.5.4 AAP_CTRL - 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																																0
Access																																RW
Name																																SYSBUSSTALL

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	SYSBUSSTALL	0	RW	**Stall the System Bus**
	When this bit is set, the system bus is stalled. Only the Cortex registers are accessible			

5.5.5 AAP_CRCCMD - CRC 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																																0
Access																																W1
Name																																CRCREQ

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	CRCREQ	0	W1	**CRC Request**
	A CRC request is generated when set to 1. This command is not available if debug access or AAP is locked.			

5.5.6 AAP_CRCSTATUS - CRC 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																																0
Access																																R
Name																																CRCBUSY

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	CRCBUSY	0	R	**CRC Calculation is Busy**
	Set when the CRC calculation is executing. Will transition from 1 to 0 on valid data.			

5.5.7 AAP_CRCADDR - CRC Address 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	0x00000000																															
Access	RW																															
Name	CRCADDR																															

Bit	Name	Reset	Access	Description
31:0	CRCADDR	0x00000000	RW	**Starting Page Address for CRC Execution**
	Set this to the address the CRC executes on.			

5.5.8 AAP_CRCRESULT - CRC Result 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	0x00000000																															
Access	R																															
Name	CRCRESULT																															

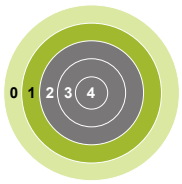
Bit	Name	Reset	Access	Description
31:0	CRCRESULT	0x00000000	R	**CRC Result of the CRCADDRESS**
	Result of the CRC calculation using the CRCADDRESS.			

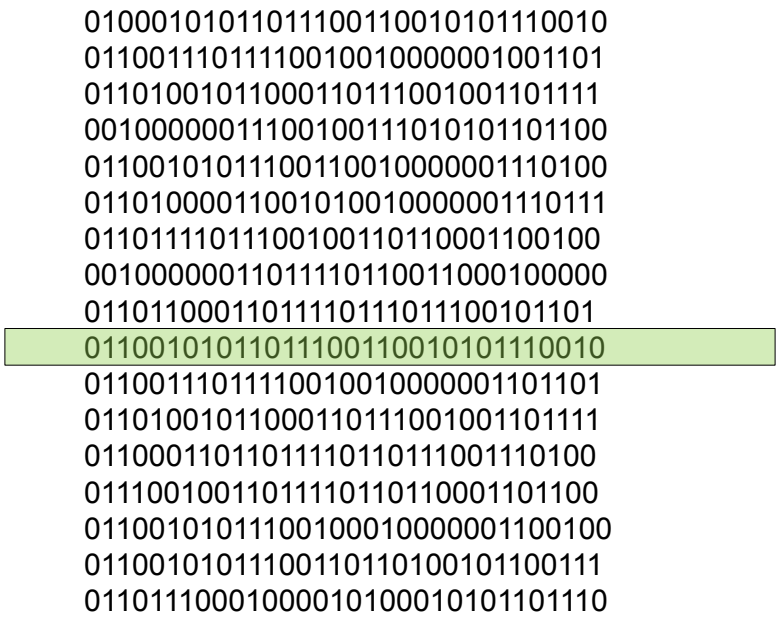
5.5.9 AAP_IDR - AAP Identification Register

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	0x26E60011																															
Access	R																															
Name	ID																															

Bit	Name	Reset	Access	Description
31:0	ID	0x26E60011	R	**AAP Identification Register**
	Access port identification register in compliance with the ARM ADI v5 specification (JEDEC Manufacturer ID) .			

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) .

Why?

The MSC allows the application code, user data and flash lock bits to be stored in non-volatile flash memory. Certain memory system functions, such as program memory wait-states and bus faults are also 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 enables minimum energy consumption by integrating low-energy flash with a charge pump so that a dedicated external supply is not needed for program and erase operations. An easy to use write and erase interface is supported by an internal, fixed-frequency oscillator and automated flash timing and control logic that reduces software complexity without using other timer resources.

Application code may dynamically scale between high energy optimization and high code execution performance through advanced read modes.

A highly efficient instruction cache reduces the number of flash reads significantly, thus saving energy. This holds true even at high operating frequencies because instruction cache hits eliminate otherwise necessary wait states. Built-in performance counters can be used to measure the efficiency of the instruction cache.

6.1 Introduction

The Memory System Controller (MSC) is the program memory unit of the EFM32 Giant Gecko 12 microcontroller. The flash memory is readable and writable from both the Cortex-M4 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. Additionally, the information block is available for special user data and flash lock bits. There is also a read-only page in the information block containing system and device calibration data, and bootloader. Read and write operations are supported in the energy modes EM0 Active and EM1 Sleep.

6.2 Features

- AHB read interface
  - Scalable access performance to optimize the Cortex-M4 code interface
    - Zero wait-state access up to 18 MHz
    - Advanced energy optimization functionality
      - Conditional branch target prefetch suppression
      - Cortex-M4 disfolding of if-then (IT) blocks
      - Instruction Cache
  - DMA read support in EM0 Active and EM1 Sleep
- Command and status interface
  - Flash write and erase
    - Accessible from Cortex-M4 in EM0 Active
    - DMA write support in EM0 Active and EM1 Sleep
    - Read-while-write support on parts with 512 kB or more flash
  - 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
  - Configurable interrupt erase abort
    - Improved interrupt predictability
  - Memory and bus fault control
- Security features
  - Lockable debug access
  - Page lock bits
  - SW mass erase lock bits
  - Authentication Access Port (AAP) lock bits
- End-of-write and end-of-erase interrupts

6.3 Functional Description

The size of the main block is device dependent. The largest size available is 1024 KB (512 pages). The information block has 2 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 169 outlines how the flash is mapped in the memory space. All flash memory is organized into 2 KB pages.

Table 6.1. MSC Flash Memory Mapping

Block	Page	Base address	Write/Erase by...	Software Readable?	Purpose/Name	Size
Main [1]	0	0x00000000	Software, debug	Yes	User code and data	256 KB - 1024 KB
	...		Software, debug	Yes		
	511	0x000FF800	Software, debug	Yes		
Reserved	-	0x00100000	-	-	Reserved for flash expansion	~24 MB
Information	0	0x0FE00000	Software, debug	Yes	User Data (UD)	2 kB
	-	0x0FE00800	-	-	Reserved	-
	1	0x0FE04000	Write: Software, debug Erase: Debug only	Yes	Lock Bits (LB)	2 kB
	-	0x0FE04800	-	-	Reserved	-
	2	0x0FE081B0	-	Yes	Device Information (DI)	2 kB
	-	0x0FE08800	-	-	Reserved	-
	3	0x0FE0C000	-	-		2 kB
	-	0x0FE0C800	-	-	Reserved	-
	4	0x0FE10000	Software, debug	Yes	Bootloader (BL)	40 kB
	...		-	-		
	23	0x0FE19800	-	-		
Reserved	-	0x0FE1A000	-	Reserved for flash expansion	Rest of code space	-

Note:
1. Block/page erased by a device erase.

6.3.1 User Data (UD) Page Description

This is the user data page in the information block. The page can be erased and written by software. The page is erased by the ERASEPAGE command of the MSC_WRITECMD register. Note that the page is not erased by a device erase operation. The device erase operation is described in 5.3.4 Authentication Access Point.

6.3.2 Lock Bits (LB) Page Description

This page contains the following information:

- Main block Page Lock Words (PLWs)
- User data page Lock Word (ULWs)
- Debug Lock Word (DLW)
- Mass erase Lock Word (MLW)
- Authentication Access Port (AAP) lock word (ALW)
- Bank switch disable (CLW1)
- Bootloader enable (CLW0)
- Pin reset soft (CLW0)

The words in this page are organized as shown in Table 6.2 Lock Bits Page Structure on page 170:

Table 6.2. Lock Bits Page Structure

127	DLW
126	ULW
125	MLW
124	ALW
123	CLW1
122	CLW0
N	PLW[N]
…	…
1	PLW[1]
0	PLW[0]

There are 32 page lock bits per page lock word (PLW). Bit 0 refers to the first page and bit 31 refers to the last page within a PLW. Thus, PLW[0] contains lock bits for page 0-31 in the main block, PLW[1] contains lock bits for page 32-63 etc. A page is locked when the bit is 0. A locked page cannot be erased or written.
Word 127 is the debug lock word (DLW). The four LSBs of this word are the debug lock bits. If these bits are 0xF, then debug access is enabled. Debug access to the core is disabled from power-on reset until the DLW is evaluated immediately before the Cortex-M4 starts execution of the user application code. If the bits are not 0xF, then debug access to the core remains blocked.

Word 126 is the user page lock word (ULW). Bit 0 of this word is the User Data Page lock bit. Bit 1 in this word locks the Lock Bits Page. The lock bits can be reset by a device erase operation initiated from the Authentication Access Port (AAP) registers. The AAP is described in more detail in 5.3.4 Authentication Access Point. Note that the AAP is only accessible from the debug interface, and cannot be accessed from the Cortex-M4 core.

Word 125 is the mass erase lock word (MLW). For devices that support read-while-write, bit 0 locks the lower half of the flash, preventing mass erase, and bit 1 locks the upper half of the flash. For devices that do not support read-while-write, bit 0 locks the entire flash. The mass erase lock bits will not have any effect on device erases initiated from the Authenitcation Access Port (AAP) registers. The AAP is described in more detail in 5.3.4 Authentication Access Point.

Word 124 is the Authentication Access Port (AAP) lock word (ALW) and the four LSBs of this word are the lock bits. If these bits are 0xF, then AAP access is enabled. If the bits are not 0xF, AAP is disabled and it is impossible to access the device through the AAP. Bit 31 of the ALW may be used to allow AAP access under controlled conditions. If bit 31 is set to 1, software running on the device can unlock AAP access using the MSC_AAPUNLOCKCMD register. If bit 31 is cleared to 0, software will not be able to use MSC_AAPUNLOCKCMD to unlock AAP access. **NOTE - locking the AAP completely (including the LSBs and bit 31) is irreversible. Once the AAP is locked, it will be impossible to perform an external mass erase and the AAP lock cannot be reset.** The only way to program the device when the AAP is locked is through a bootloader or by SW already loaded into the FLASH.

Word 123 is Configuration Lock Word 1 (CLW1). Bit 0 is the Bank Switch Disable bit. Because the state of erased flash bits is 1, bank switching is disabled on dual-bank devices by default. To enable bank switching, this bit must be written to 0.

Word 122 is Configuration Lock Word 0 (CLW0). Bit 2 is the Pin Reset Soft bit. By default, a pin reset is handled as a soft reset (See 8.3.5 RESETn Pin Reset). Bit 1 is the bootloader enable bit. Because the state of erased flash bits is 1, the bootloader is enabled by default.

Note: A hard reset is required in order for changes to the lock bits to take effect. In typical use cases, this will be a power-on reset because pin reset is a soft reset by default (see above).

6.3.3 Device Information (DI) Page

This read-only page holds calibration data from the production test as well as a unique device ID. The page is further described in 4. Memory and Bus System.

6.3.4 Bootloader

The system is configured by default to boot from a pre-programmed bootloader automatically after system reset. The bootloader is described in *AN0003: UART Bootloader* (www.silabs.com/32bit-appnotes). Users can bypass the bootloader by clearing bit 1 in Configuration Lock Word 0 (CLW0) at word 122 in the lock bits page.

After any device reset, the bootloader area is accessible to both software reads and writes. Reading and writing of this area may be disabled with the MSC_BOOTLOADERCTRL register. Note that this register is write-once, so after writing the register, a reset of the system is required in order to change permissions again.

The bootloader size is 40 kB for this device family.

Note: Software should never erase "Reserved" pages when bootloader write/erase is enabled. Doing so may cause the device to become non-functional and irrevocably locked.

6.3.5 Post-Reset Behavior

Calibration values are automatically written to registers by the MSC before application code startup. The values are also available to read from the DI page for later reference by software. Other information such as the device ID and production date is also stored in the DI page and is readable from software.

If the bootloader is not bypassed, the system will boot up from the bootloader at address 0x0FE10000.

6.3.6 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 startup-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 Cortex will stall on the first access to the flash until it is ready. Execute code from RAM or cache to get a quicker startup.

To get the fastest possible startup when waking, i.e. a startup that depends on the current operating conditions, set STDLY0 to 0x28 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 even quicker startup, run code in beginning with a set of wait states. 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, sampling will begin with the given number of wait states after 5 us, and the system will run with this number of wait states for the remaining 5 us before returning to normal operation

A recommended setting for MSC_STARTUP register is to leave STDLY0 at its reset value and set ASTWAIT to one for active sampling Set STWSEN to zero to bypass the second delay period.

Flash wakeup on demand is supported when wakeup from EM2/3 to EM0. Set bit PWRUPONDEMAND of register MSC_CTRL to one to enable the power up on demand. When enabled during powerup, flash will enter sleep mode and waiting for either pending flash read transaction or software command to MSC_CMD.PWRUP bit. If software command wakeup, and interrupt of MSC_IF.PWRUPF will be flaged if the MSC_IEN.PWRUPF is set

6.3.7 Wait States

Table 6.3. Flash Wait States

Wait States	Frequency
WS0	no more than 18 MHz
WS1	above 18 MHz and no more than 36 MHz
WS2	above 36 MHz and no more than 54 MHz
WS3	above 54 MHz and no more than 72 MHz

Along with the flash wait state configuration, users must set the Peripheral Access Wait Mode (WAITMODE in MSC_CTRL), according to specified operating mode limits.

6.3.7.1 One Wait State Access

After reset, the HFCORECLK is normally 19 MHz from the HFRCO and the MODE field of the MSC_READCTRL register is set to WS1 (one wait-state). Software must not select a zero wait-state mode unless the clock is guaranteed to be 18 MHz or below, otherwise the resulting behavior is undefined. If a HFCORECLK frequency above 18 MHz is to be set by software, the MODE field of the MSC_READCTRL register must be set to WS1 or WS1SCBTP before the core clock is switched to the higher frequency clock source.

When changing to a lower frequency, the MODE field of the MSC_READCTRL register must be set to WS0 or WS0SCBTP only after the frequency transition has completed. If the HFRCO is used, wait until the oscillator is stable on the new frequency. Otherwise, the behavior is unpredictable.

6.3.7.2 Zero Wait-state Access

At 18 MHz and below, read operations from flash may be performed without any wait states. Zero wait state access greatly improves code execution performance at frequencies of 18 MHz and lower. By default, the Cortex-M4 uses speculative prefetching and if-then block folding to maximize code execution performance at the cost of additional flash accesses and energy consumption.

6.3.8 Suppressed Conditional Branch Target Prefetch (SCBTP)

MSC offers a special instruction fetch mode which optimizes energy consumption by cancelling Cortex-M4 conditional branch target prefetches. Normally, the Cortex-M4 core prefetches both the next sequential instruction and the instruction at the branch target address when a conditional branch instruction reaches the pipeline decode stage. This prefetch scheme improves performance while one extra instruction is fetched from memory at each conditional branch, regardless of whether the branch is taken or not. To optimize for low energy, the MSC can be configured to cancel these speculative branch target prefetches. With this configuration, energy consumption is more optimal, as the branch target instruction fetch is delayed until the branch condition is evaluated.

The performance penalty with this mode enabled is source code dependent, but is normally less than 1% for core frequencies from 18 MHz and below. To enable the mode at frequencies from 18 MHz and below write WS0SCBTP to the MODE field of the MSC_READCTRL register. For frequencies above 18 MHz, use the WS1SCBTP mode, and for frequencies above 36 MHz, use the WS2SCBTP mode. An increased performance penalty per clock cycle must be expected compared to WS0SCBTP mode. The performance penalty in WS1SCBTP/WS2SCBTP mode depends greatly on the density and organization of conditional branch instructions in the code.

6.3.9 Cortex-M4 If-Then Block Folding

The Cortex-M4 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-M4 Technical Reference Manual for details. Normally, it is expected that this feature is most efficient at core frequencies above 18 MHz. Folding is enabled by default.

6.3.10 Instruction Cache

The MSC includes an instruction cache. The instruction cache for the internal flash memory is enabled by default, but can be disabled by setting IFCDIS in MSC_READCTRL. 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. Thus, performance is also improved when wait-states are used (i.e. running at frequencies above 18 MHz).

The instruction cache is connected directly to the ICODE 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 on page 173. The cache consists of an access filter, lookup logic, SRAM, and two performance counters. The access filter checks that the address for the access is to on-chip flash memory (instructions in RAM are not cached). If the address matches, the cache lookup logic and SRAM is enabled. Otherwise, the cache is bypassed and the access is forwarded to the memory system. The cache is then updated when the memory access completes. The access filter also disables cache updates for interrupt context accesses if caching in interrupt context is disabled. The performance counters, when enabled, keep track of the number of cache hits and misses. The cachelines are filled up continuously one word at a time as the individual words are requested by the processor. Thus, not all words of a cacheline might be valid at a given time.

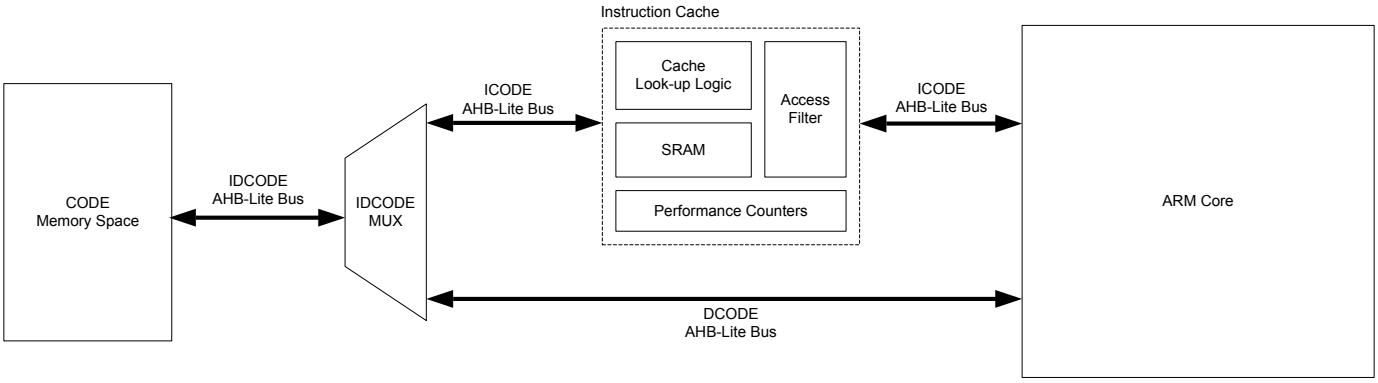


Figure 6.1. Instruction Cache

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 AIDIS in MSC_READCTRL. The cache can (independent of the AIDIS setting) be manually invalidated by writing 1 to INVCACHE in MSC_CMD.

Note: The instruction cache flush is not triggered at the event of a bus fault. As a result, when an instruction fetch results in a bus fault, invalid data may be cached. This means that the next time the instruction that caused the bus fault is fetched, the processor core will get the invalid cached data without any bus fault. In order to avoid invalid cached data propagation to the processor core, software should manually invalidate the instruction cache by writing 1 to INVCACHE in MSC_CMD at the event of a bus fault.

In general it is highly recommended to keep the cache enabled all the time. However, for some sections of code with very low cache hit-rate more energy-efficient execution can be achieved by disabling the cache temporarily. 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 writing 1 to STARTPC in MSC_CMD. This starts the performance counters, counting from 0. At the end of the section, stop the performance counters by writing 1 to STOPPC in MSC_CMD. The number of cache hits and cache misses for that section can then be read from MSC_CACHEHITS and MSC_CACHEMISSES respectively. The total number of 32-bit instruction fetches will be MSC_CACHEHITS + MSC_CACHEMISSES. Thus, the cache hit-ratio can be calculated as MSC_CACHEHITS / (MSC_CACHEHITS + MSC_CACHEMISSES). When MSC_CACHEHITS overflows the CHOF interrupt flag is set. When MSC_CACHEMISSES overflows the CMOF interrupt flag is set. These flags must be cleared explicitly by software. The range of the performance counters can thus be extended by increasing a counter in the MSC interrupt routine. The performance counters only count when a cache lookup is performed. If the lookup fails, MSC_CACHEMISSES is increased. If the lookup is successful, MSC_CACHEHITS is increased. For example, a cache lookup is not performed if the cache is disabled or the code is executed from RAM.

Note: When caching of vector fetches and instructions in interrupt routines is disabled (ICCDIS in MSC_READCTRL is set), the performance counters do not count when these types of fetches occur (i.e. while in interrupt context).

By default, interrupt vector fetches and instructions in interrupt routines are also cached. Some applications may get better cache utilization by not caching instructions in interrupt context. This is done by setting ICCDIS in MSC_READCTRL. You should only set this bit based on the results from a cache hit ratio measurement. In general, it is recommended to keep the ICCDIS bit cleared. Note that lookups in the cache are still performed, regardless of the ICCDIS setting - but instructions are not cached when cache misses occur inside

the interrupt routine. So, for example, if a cached function is called from the interrupt routine, the instructions for that function will be taken from the cache.

The cache content is not retained in EM2, EM3 and EM4. The cache is therefore invalidated regardless of the setting of AIDIS in MSC_READCTRL when entering these energy modes. Applications that switch frequently between EM0 and EM2/3 and executes the very same non-looping code almost every time will most likely benefit from putting this code in RAM. The interrupt vectors can also be put in RAM to reduce current consumption even further.

The cache also supports caching of instruction fetches from the external bus interface (EBI) when accessing the EBI through code space. By default, this is enabled, but it can be disabled by setting EBICDIS in MSC_READCTRL.

6.3.11 Low Voltage Flash Read

The devices support low voltage flash reads. Because it takes more time to read from flash with a lower voltage supply MSC_READCTRL.MODE should be programmed accordingly. It is recommended that software should follow certain sequences for supply voltage scaling up and down. See the EMU chapter for details.

Flash write/erase is not supported in low voltage mode. Any write/erase command will be ignored if flash is operated in a low voltage mode and the interrupt flag MSC_IF.LVEWRITE will be set.

6.3.12 Bank Switching Operation

It is possible to swap the starting addresses of the two flash instances under software control by issuing a bank switch command. The BANKSWITCHED bit in the MSC_STATUS register indicates if address 0x0 currently points to flash instance 0 or flash instance 1. After power-on reset, flash instance 0 always starts at address 0x0. Writing a 1 to the SWITCHINGBANK bit in the MSC_CMD register swaps the two flash instances immediately.

The bank switch command must be executed from the bootloader region of flash or RAM, as doing so from one of the main flash instances can result in unpredictable behavior.

Note: Bank switching is disabled unless bit 0 of Configuration Lock Word 1 has been programmed to 0 prior to exit from power-on reset.

A typical use of bank switching is described here: During the chip power up sequence, the bootloader fetches parameters from fixed addresses of both flash instances. The parameters could be firmware revision numbers, error checking codes, or other flags. Based on these parameters, the bootloader determines whether or not bank switching is necessary. The bank switching operation must be executed before the CPU transfers execution control to the application in the selected flash instance.

6.3.13 Erase and Write Operations

Both page erase and write operations require that the address is written into the MSC_ADDRB register. For erase operations, the address may be any within the page to be erased. Load the address by writing 1 to the LADDRIM bit in the MSC_WRITECMD register. The LADDRIM bit only has to be written once when loading the first address. After each word is written the internal address register ADDR will be incremented automatically by 4. The INVADDR bit of the MSC_STATUS register is set if the loaded address is outside the flash and the LOCKED bit of the MSC_STATUS register is set if the page addressed is locked. Any attempts to command erase of or write to the page are ignored if INVADDR or the LOCKED bits of the MSC_STATUS register are set. To abort an ongoing erase, set the ERASEABORT bit in the MSC_WRITECMD register.

When a word is written to the MSC_WDATA register, the WDATAREADY bit of the MSC_STATUS register is cleared. When this status bit is set, software or DMA may write the next word.

A single word write is commanded by setting the WRITEONCE bit of the MSC_WRITECMD register. The operation is complete when the BUSY bit of the MSC_STATUS register is cleared and control of the flash is handed back to the AHB interface, allowing application code to resume execution.

For a DMA write the software must write the first word to the MSC_WDATA register and then set the WRITETRIG bit of the MSC_WRITECMD register. DMA triggers when the WDATAREADY bit of the MSC_STATUS register is set.

It is possible to write words twice between each erase by keeping at 1 the bits that are not to be changed. Let us take as an example writing two 16 bit values, 0xAAAA and 0x5555. To safely write them in the same flash word this method can be used:

- Write 0xFFFFAAAA (word in flash becomes 0xFFFFAAAA)
- Write 0x5555FFFF (word in flash becomes 0x5555AAAA)

Note:

- There is a maximum of two writes to the same word between each erase due to a physical limitation of the flash.
- Flash write/erase is not supported in low voltage mode. Any write/erase command will be ignored if flash is operated in a low voltage mode and the interrupt flag MSC_IF.LVEWRITE will be set.
- During a write or erase, flash read accesses not subject to read-while-write will be stalled, effectively halting code execution from flash. Code execution continues upon write/erase completion. Code residing in RAM may be executed during a write/erase operation regardless of whether read-while-write is enabled or not.

6.3.13.1 Read-While-Write

Parts with 512 kB or 1 MB of flash support read-while-write operation. For example, it is possible for the CPU to execute (read) code from the lower half of the flash while programming or erasing the upper half of the flash and vice versa. Enable read-while-write by setting the RWWEN bit in the MSC_WRITECTRL register.

The physical flash pages allocated to the lock bits, user data, and bootloader regions are interleaved between the two flash instances as shown below:

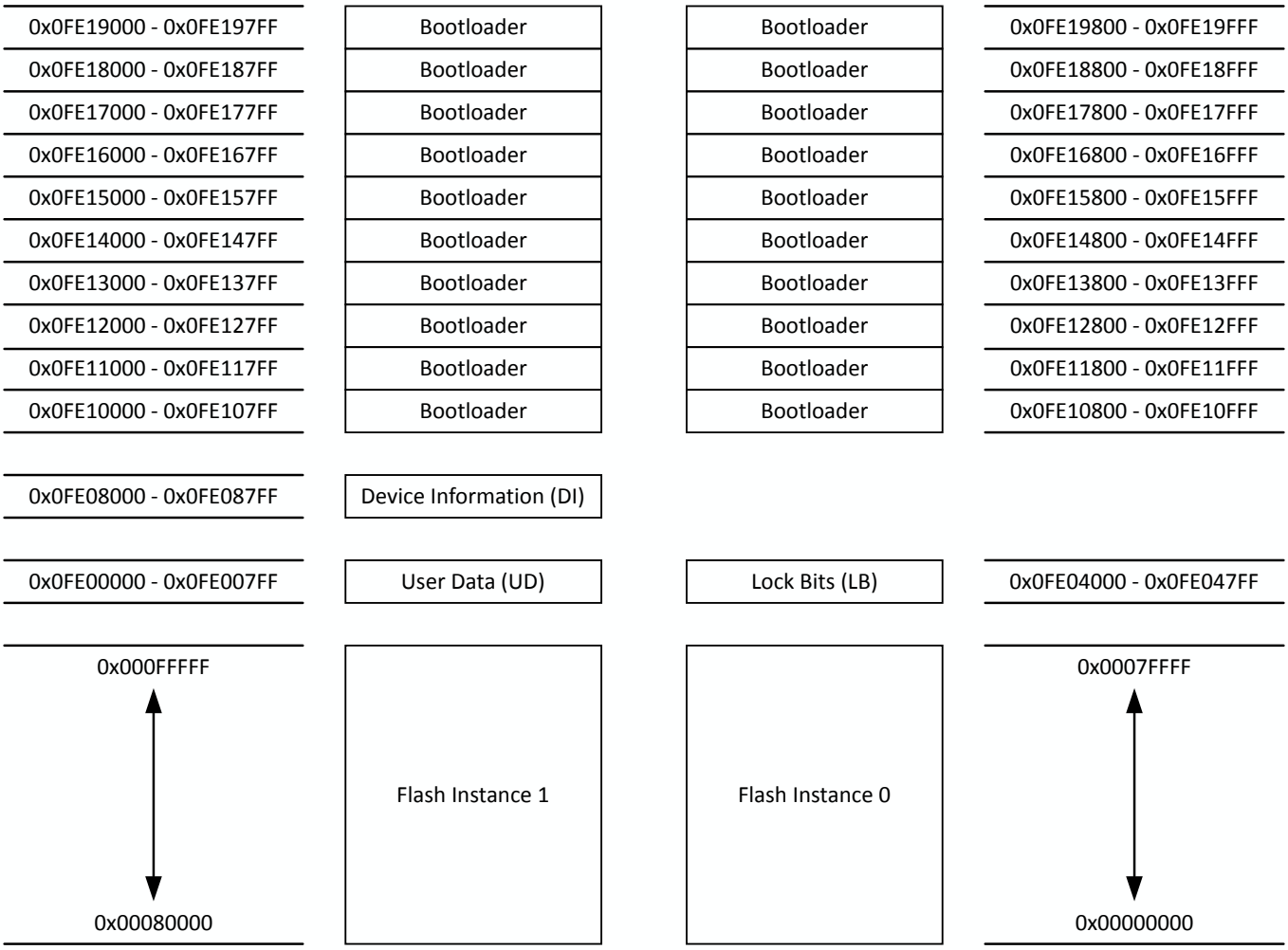


Figure 6.2. Flash Page Organization and Instance Association

The address ranges for flash instances 0 and 1 in Figure 6.2 Flash Page Organization and Instance Association on page 176 are for a 1 MB device. On a 512 kB device, flash instances 0 and 1 are half the size (256 kB vs. 512 kB) and are mapped starting at 0x0 and 0x40000, respectively, in order provide a contiguous address space. The arrangement of the lock bits, user data, and bootloader pages is the same in both cases.

6.3.13.2 Mass Erase

A mass erase can be initiated from software using ERASEMAIN0 and ERASEMAIN1 in MSC_WRITECMD. These commands will for devices supporting read-while-write start a masserase on the lower and upper half of the flash respectively. For devices not supporting read-while-write, MASSERASE0 will start a mass erase of the entire flash. Prior to initiating a mass erase, MSC_MASSLOCK must be unlocked by writing 0x631A to it. After a mass erase has been started, this register can be locked again to prevent runaway code from accidentally triggering a mass erase.

The regular flash page lock bits will not prevent a mass erase. To prevent software from initiating mass erases, use the mass erase lock bits in the mass erase lock word (MLW).

6.4 Register Map

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

Offset	Name	Type	Description
0x000	MSC_CTRL	RWH	Memory System Control Register
0x004	MSC_READCTRL	RWH	Read Control Register
0x008	MSC_WRITECTRL	RW	Write Control Register
0x00C	MSC_WRITECMD	W1	Write Command Register
0x010	MSC_ADDRB	RW	Page Erase/Write Address Buffer
0x018	MSC_WDATA	RW	Write Data Register
0x01C	MSC_STATUS	R	Status Register
0x030	MSC_IF	R	Interrupt Flag Register
0x034	MSC_IFS	W1	Interrupt Flag Set Register
0x038	MSC_IFC	(R)W1	Interrupt Flag Clear Register
0x03C	MSC_IEN	RW	Interrupt Enable Register
0x040	MSC_LOCK	RWH	Configuration Lock Register
0x044	MSC_CACHECMD	W1	Flash Cache Command Register
0x048	MSC_CACHEHITS	R	Cache Hits Performance Counter
0x04C	MSC_CACHEMISSES	R	Cache Misses Performance Counter
0x054	MSC_MASSLOCK	RWH	Mass Erase Lock Register
0x05C	MSC_STARTUP	RW	Startup Control
0x070	MSC_BANKSWITCHLOCK	RWH	Bank Switching Lock Register
0x074	MSC_CMD	W1	Command Register
0x090	MSC_BOOTLOADERCTRL	RW	Bootloader Read and Write Enable, Write Once Register
0x094	MSC_AAPUNLOCKCMD	W1	Software Unlock AAP Command Register
0x098	MSC_CACHECONFIG0	RW	Cache Configuration Register 0
0x100	MSC_RAMCTRL	RW	RAM Control Enable Register
0x104	MSC_ECCCTRL	RW	RAM ECC Control Register
0x108	MSC_RAMECCADDR	R	RAM ECC Error Address Register
0x10C	MSC_RAM1ECCADDR	R	RAM1 ECC Error Address Register
0x110	MSC_RAM2ECCADDR	R	RAM2 ECC Error Address Register

6.5 Register Description

6.5.1 MSC_CTRL - Memory System 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																				0						0	1	0	0	0	0	1
Access																				RWH						RW	RW	RW	RW	RW	RW	RW
Name																				WAITMODE						EBIFAULTEN	RAMECCERRFAULTEN	TIMEOUTFAULTEN	IFCREADCLEAR	PWRUPONDEMAND	CLKDISFAULTEN	ADDRFAULTEN

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	WAITMODE	0	RWH	**Peripheral Access Wait Mode**

This field controls the wait-state configuration for peripheral accesses over the bus. Any time software wants to increase the bus clock frequency or decrease the voltage, this register must be prior set to a valid value for the new operating mode. When changing to a lower frequency or higher voltage, this register can be set to a valid value for the new operating mode after the transition has been completed. If the HFRCO is used as clock source, wait until the oscillator is stable on the new frequency to avoid unpredictable behavior. In the mode descriptions below, the number of 'cycles latency' is in terms of bus clk cycles does not include any additional delays added due to prescaled slave or master clocks.

Value	Mode	Description
0	WS0	One cycle latency for reads. Zero cycles latency for writes. This mode or higher is required always.
1	WS1	One cycle latency for reads. Two cycles latency for writes. This mode is required for a bus frequency above 50 MHz (at 1.2V).

11:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	EBIFAULTEN	0	RW	**EBI Bus Fault Response Enable**

When this bit is set, bus faults are generated when EBI has invalid transcations

5	RAMECCERRFAULTEN	1	RW	**Two Bit ECC Error Bus Fault Response Enable**

When this bit is set, bus faults are generated if 2-bit ECC error occurs

4	TIMEOUTFAULTEN	0	RW	**Timeout Bus Fault Response Enable**

When this bit is set, bus faults are generated when the bus system times out during an access, e.g., when reading a register from an LE peripheral that is changing too fast to get a stable value

3	IFCREADCLEAR	0	RW	**IFC Read Clears IF**

This bit controls what happens when an IFC register in a module is read.

Value	Description

Bit	Name	Reset	Access	Description
	0			IFC register reads 0. No side-effect when reading.
	1			IFC register reads the same value as IF, and the corresponding interrupt flags are cleared.
2	PWRUPONDEMAND	0	RW	**Power Up on Demand During Wake Up** When set, during wake up, pending AHB transfer will cause MSC to issue power up request to CMU. If not set, will always issue power up request if PWRUPONCMD is not set either.
1	CLKDISFAULTEN	0	RW	**Clock-disabled Bus Fault Response Enable** When this bit is set, busfaults are generated on accesses to peripherals/system devices with clocks disabled
0	ADDRFAULTEN	1	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

6.5.2 MSC_READCTRL - Read 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				0			0x1															0	0	1		0	0	0	0			
Access				RW			RWH															RW	RW	RW		RW	RW	RW	RW			
Name				SCBTP			MODE															QSPICDIS	USEHPROT	PREFETCH		EBICDIS	ICCDIS	AIDIS	IFCDIS			

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	SCBTP	0	RW	**Suppress Conditional Branch Target Perfetch**
	Enable suppressed Conditional Branch Target Prefetch (SCBTP) function. SCBTP saves energy by delaying Cortex-M conditional branch target prefetches until the conditional branch instruction is in the execute stage. When the instruction reaches this stage, the evaluation of the branch condition is completed and the core does not perform a speculative prefetch of both the branch target address and the next sequential address. With the SCBTP function enabled, one instruction fetch is saved for each branch not taken, with a negligible performance penalty.			
27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:24	MODE	0x1	RWH	**Read Mode**
	After reset, the core clock is 19 MHz from the HFRCO and the MODE field of MSC_READCTRL register is set to WS1. The reset value is WS1 because the HFRCO may produce a frequency above 19 MHz before it is calibrated. A large wait states is associated with high frequency. When changing to a higher frequency, this register must be set to a large wait states first before the core clock is switched to the higher frequency. When changing to a lower frequency, this register should be set to lower wait states after the frequency transition has been completed. If the HFRCO is used as clock source, wait until the oscillator is stable on the new frequency to avoid unpredictable behavior.See Flash Wait-States table for the corresponding threshold for different wait-states.			

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. See Flash Wait-States table for details
2	WS2	Two wait-states inserted for eatch fetch or read transfer. See Flash Wait-States table for details
3	WS3	Three wait-states inserted for eatch fetch or read transfer. See Flash Wait-States table for details

23:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	QSPICDIS	0	RW	**QSPI Cache Disable**
	Disable instruction cache for QSPI.			
9	USEHPROT	0	RW	**AHB_HPROT Mode**
	Use ahb_hrpot to determine if the instruction is cacheable or not			
8	PREFETCH	1	RW	**Prefetch Mode**
	Set to configure level of prefetching.			

Bit	Name	Reset	Access	Description
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	EBICDIS	0	RW	**External Bus Interface Cache Disable**
	Disable instruction cache for external bus interface.			
5	ICCDIS	0	RW	**Interrupt Context Cache Disable**
	Set this bit to automatically disable caching of vector fetches and instruction fetches in interrupt context. Cache lookup will still be performed in interrupt context. When set, the performance counters will not count when these types of fetches occur.			
4	AIDIS	0	RW	**Automatic Invalidate Disable**
	When this bit is set the cache is not automatically invalidated when a write or page erase is performed.			
3	IFCDIS	0	RW	**Internal Flash Cache Disable**
	Disable instruction cache for internal flash memory.			
2:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

6.5.3 MSC_WRITECTRL - Write 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																											0				0	0
Access																											RW				RW	RW
Name																											RWWEN				IRQERASEABORT	WREN

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	RWWEN	0	RW	**Read-While-Write Enable**
	When set, reads to the upper half of the flash can be done while writes/erases are being done in the lower half of the flash, and vice versa. Reading from the same half as a flash write/erase will stall the access until the write/erase has completed.			
4:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	IRQERASEABORT	0	RW	**Abort Page Erase on Interrupt**
	When this bit is set to 1, any Cortex-M interrupt aborts any current page erase operation. Executing that interrupt vector from Flash will halt the CPU.			
0	WREN	0	RW	**Enable Write/Erase Controller**
	When this bit is set, the MSC write and erase functionality is enabled			

6.5.4 MSC_WRITECMD - Write 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																				0			0	0			0	0	0	0	0	0
Access																				W1			W1	W1			W1	W1	W1	W1	W1	W1
Name																				CLEARWDATA			ERASEMAIN1	ERASEMAIN0			ERASEABORT	WRITETRIG	WRITEONCE	WRITEEND	ERASEPAGE	LADDRIM

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	0	W1	**Clear WDATA State**
	Will set WDATAREADY and DMA request. Should only be used when no write is active.			
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	ERASEMAIN1	0	W1	**Mass Erase Region 1**
	Initiate mass erase of region 1. For devices supporting read-while-write, this is the upper half of the flash. Before use MSC_MASSLOCK must be unlocked. To completely prevent access from software, clear bit 1 in the mass erase lock-word (MLW)			
8	ERASEMAIN0	0	W1	**Mass Erase Region 0**
	Initiate mass erase of region 0. For devices supporting read-while-write, this is the lower half of the flash. For other devices it is the entire flash. 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	0	W1	**Abort Erase Sequence**
	Writing to this bit will abort an ongoing erase sequence.			
4	WRITETRIG	0	W1	**Word Write Sequence Trigger**
	Start write of the first word written to MSC_WDATA, then add 4 to ADDR and write the next word if available within a 30us timeout. When ADDR is incremented past the page boundary, ADDR is set to the base of the page. If WDOUBLE is set, two words are required every time, and ADDR is incremented by 8.			
3	WRITEONCE	0	W1	**Word Write-Once Trigger**
	Write the word in MSC_WDATA to ADDR. Flash access is returned to the AHB interface as soon as the write operation completes. The WREN bit in the MSC_WRITECTRL register must be set in order to use this command. Only a single word is written, but the internal address is also incremented to allow a direct write of a new word without loading a new address			
2	WRITEEND	0	W1	**End Write Mode**
	Write 1 to end write mode when using the WRITETRIG command.			
1	ERASEPAGE	0	W1	**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.			

Bit	Name	Reset	Access	Description
0	LADDRIM	0	W1	**Load MSC_ADDRB Into ADDR**
	Load the internal write address register ADDR from the MSC_ADDRB register. The internal address register ADDR is incremented automatically by 4 after each word is written. When ADDR is incremented past the page boundary, ADDR is set to the base of the page.			

6.5.5 MSC_ADDRB - Page Erase/Write Address Buffer

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	0x00000000																															
Access	RW																															
Name	ADDRB																															

Bit	Name	Reset	Access	Description
31:0	ADDRB	0x00000000	RW	**Page Erase or Write Address Buffer**
	This register holds the page address for the erase or write operation. This register is loaded into the internal MSC_ADDR register when the LADDRIM field in MSC_WRITECMD is set.			

6.5.6 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	0x00000000																															
Access	RW																															
Name	WDATA																															

Bit	Name	Reset	Access	Description
31:0	WDATA	0x00000000	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.			

6.5.7 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				0x0																				0	0	0	0	1	0	0	0
Access	R				R																				R	R	R	R	R	R	R	R
Name	PWRUPCKBDFAILCOUNT				WDATAVALID																				BANKSWITCHED	PCRUNNING	ERASEABORTED	WORDTIMEOUT	WDATAREADY	INVADDR	LOCKED	BUSY

Bit	Name	Reset	Access	Description
31:28	PWRUPCKBDFAIL-COUNT	0x0	R	**Flash Power Up Checkerboard Pattern Check Fail Count**
	This field tells how many times checkboard pattern check fail occured after a reset sequence.			
27:24	WDATAVALID	0x0	R	**Write Data Buffer Valid Flag**
	This field tells how many valid data in the write buffer, each bit indicates one buffer entry			
23:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	BANKSWITCHED	0	R	**BANK SWITCHING STATUS**
	This bit is zero while the address 0 starts from region 0. This bit is one when address 0 starts from region 1.			
6	PCRUNNING	0	R	**Performance Counters Running**
	This bit is set while the performance counters are running. When one performance counter reaches the maximum value, this bit is cleared.			
5	ERASEABORTED	0	R	**The Current Flash Erase Operation Aborted**
	When set, the current erase operation was aborted by interrupt.			
4	WORDTIMEOUT	0	R	**Flash Write Word Timeout**
	When this bit is set, MSC_WDATA was not written within the timeout. The flash write operation timed out and access to the flash is returned to the AHB interface. This bit is cleared when the ERASEPAGE, WRITETRIG or WRITEONCE commands in MSC_WRITECMD are triggered.			
3	WDATAREADY	1	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	0	R	**Invalid Write Address or Erase Page**
	Set when software attempts to load an invalid (unmapped) address into ADDR			
1	LOCKED	0	R	**Access Locked**
	When set, the last erase or write is aborted due to erase/write access constraints			
0	BUSY	0	R	**Erase/Write Busy**
	When set, an erase or write operation is in progress and new commands are ignored			

6.5.8 MSC_IF - Interrupt Flag 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											0	0	0	0	0	0								0		0	0	0	0	0	0	0
Access											R	R	R	R	R	R								R		R	R	R	R	R	R	R
Name											RAM2ERR2B	RAM2ERR1B	RAM1ERR2B	RAM1ERR1B	RAMERR2B	RAMERR1B								LVEWRITE		WDATAOV	ICACHERR	PWRUPF	CMOF	CHOF	WRITE	ERASE

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	RAM2ERR2B	0	R	**RAM2 2-bit ECC Error Interrupt Flag**
	RAM2 2-bit ECC Error Interrupt flag.			
20	RAM2ERR1B	0	R	**RAM2 1-bit ECC Error Interrupt Flag**
	RAM2 1-bit ECC Error Interrupt flag.			
19	RAM1ERR2B	0	R	**RAM1 2-bit ECC Error Interrupt Flag**
	RAM1 2-bit ECC Error Interrupt flag.			
18	RAM1ERR1B	0	R	**RAM1 1-bit ECC Error Interrupt Flag**
	RAM1 1-bit ECC Error Interrupt flag.			
17	RAMERR2B	0	R	**RAM 2-bit ECC Error Interrupt Flag**
	RAM 2-bit ECC Error Interrupt flag.			
16	RAMERR1B	0	R	**RAM 1-bit ECC Error Interrupt Flag**
	RAM 1-bit ECC Error Interrupt flag.			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	LVEWRITE	0	R	**Flash LVE Write Error Flag**
	If one, flash controller write command received while in LVE mode			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	WDATAOV	0	R	**Flash Controller Write Buffer Overflow**
	If one, flash controller write buffer overflow detected			
5	ICACHERR	0	R	**ICache RAM Parity Error Flag**
	If one, iCache RAM parity Error detected			
4	PWRUPF	0	R	**Flash Power Up Sequence Complete Flag**
	Set after MSC_CMD.PWRUP received, flash powered up complete and ready for read/write			
3	CMOF	0	R	**Cache Misses Overflow Interrupt Flag**
	Set when MSC_CACHEMISSES overflows			

Bit	Name	Reset	Access	Description
2	CHOF	0	R	**Cache Hits Overflow Interrupt Flag**
	Set when MSC_CACHEHITS overflows			
1	WRITE	0	R	**Write Done Interrupt Read Flag**
	Set when a write is done			
0	ERASE	0	R	**Erase Done Interrupt Read Flag**
	Set when erase is done			

6.5.9 MSC_IFS - Interrupt Flag Set 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											0	0	0	0	0	0								0		0	0	0	0	0	0	0
Access											W1	W1	W1	W1	W1	W1								W1		W1	W1	W1	W1	W1	W1	W1
Name											RAM2ERR2B	RAM2ERR1B	RAM1ERR2B	RAM1ERR1B	RAMERR2B	RAMERR1B								LVEWRITE		WDATAOV	ICACHERR	PWRUPF	CMOF	CHOF	WRITE	ERASE

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	RAM2ERR2B	0	W1	**Set RAM2ERR2B Interrupt Flag**
	Write 1 to set the RAM2ERR2B interrupt flag			
20	RAM2ERR1B	0	W1	**Set RAM2ERR1B Interrupt Flag**
	Write 1 to set the RAM2ERR1B interrupt flag			
19	RAM1ERR2B	0	W1	**Set RAM1ERR2B Interrupt Flag**
	Write 1 to set the RAM1ERR2B interrupt flag			
18	RAM1ERR1B	0	W1	**Set RAM1ERR1B Interrupt Flag**
	Write 1 to set the RAM1ERR1B interrupt flag			
17	RAMERR2B	0	W1	**Set RAMERR2B Interrupt Flag**
	Write 1 to set the RAMERR2B interrupt flag			
16	RAMERR1B	0	W1	**Set RAMERR1B Interrupt Flag**
	Write 1 to set the RAMERR1B interrupt flag			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	LVEWRITE	0	W1	**Set LVEWRITE Interrupt Flag**
	Write 1 to set the LVEWRITE interrupt flag			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	WDATAOV	0	W1	**Set WDATAOV Interrupt Flag**
	Write 1 to set the WDATAOV interrupt flag			
5	ICACHERR	0	W1	**Set ICACHERR Interrupt Flag**
	Write 1 to set the ICACHERR interrupt flag			
4	PWRUPF	0	W1	**Set PWRUPF Interrupt Flag**
	Write 1 to set the PWRUPF interrupt flag			
3	CMOF	0	W1	**Set CMOF Interrupt Flag**
	Write 1 to set the CMOF interrupt flag			

Bit	Name	Reset	Access	Description
2	CHOF	0	W1	**Set CHOF Interrupt Flag**
	Write 1 to set the CHOF interrupt flag			
1	WRITE	0	W1	**Set WRITE Interrupt Flag**
	Write 1 to set the WRITE interrupt flag			
0	ERASE	0	W1	**Set ERASE Interrupt Flag**
	Write 1 to set the ERASE interrupt flag			

6.5.10 MSC_IFC - Interrupt Flag Clear 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											0	0	0	0	0	0								0		0	0	0	0	0	0	0
Access											(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1								(R)W1		(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name											RAM2ERR2B	RAM2ERR1B	RAM1ERR2B	RAM1ERR1B	RAMERR2B	RAMERR1B								LVEWRITE		WDATAOV	ICACHERR	PWRUPF	CMOF	CHOF	WRITE	ERASE

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	RAM2ERR2B	0	(R)W1	**Clear RAM2ERR2B Interrupt Flag**
	Write 1 to clear the RAM2ERR2B interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
20	RAM2ERR1B	0	(R)W1	**Clear RAM2ERR1B Interrupt Flag**
	Write 1 to clear the RAM2ERR1B interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
19	RAM1ERR2B	0	(R)W1	**Clear RAM1ERR2B Interrupt Flag**
	Write 1 to clear the RAM1ERR2B interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
18	RAM1ERR1B	0	(R)W1	**Clear RAM1ERR1B Interrupt Flag**
	Write 1 to clear the RAM1ERR1B interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
17	RAMERR2B	0	(R)W1	**Clear RAMERR2B Interrupt Flag**
	Write 1 to clear the RAMERR2B interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
16	RAMERR1B	0	(R)W1	**Clear RAMERR1B Interrupt Flag**
	Write 1 to clear the RAMERR1B interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	LVEWRITE	0	(R)W1	**Clear LVEWRITE Interrupt Flag**
	Write 1 to clear the LVEWRITE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	WDATAOV	0	(R)W1	**Clear WDATAOV Interrupt Flag**
	Write 1 to clear the WDATAOV interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
5	ICACHERR	0	(R)W1	**Clear ICACHERR Interrupt Flag**
	Write 1 to clear the ICACHERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
4	PWRUPF	0	(R)W1	**Clear PWRUPF Interrupt Flag**
	Write 1 to clear the PWRUPF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	CMOF	0	(R)W1	**Clear CMOF Interrupt Flag**
	Write 1 to clear the CMOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	CHOF	0	(R)W1	**Clear CHOF Interrupt Flag**
	Write 1 to clear the CHOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	WRITE	0	(R)W1	**Clear WRITE Interrupt Flag**
	Write 1 to clear the WRITE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	ERASE	0	(R)W1	**Clear ERASE Interrupt Flag**
	Write 1 to clear the ERASE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

6.5.11 MSC_IEN - Interrupt Enable 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											0	0	0	0	0	0								0		0	0	0	0	0	0	0
Access											RW	RW	RW	RW	RW	RW								RW		RW	RW	RW	RW	RW	RW	RW
Name											RAM2ERR2B	RAM2ERR1B	RAM1ERR2B	RAM1ERR1B	RAMERR2B	RAMERR1B								LVEWRITE		WDATAOV	ICACHERR	PWRUPF	CMOF	CHOF	WRITE	ERASE

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	RAM2ERR2B	0	RW	**RAM2ERR2B Interrupt Enable**
	Enable/disable the RAM2ERR2B interrupt			
20	RAM2ERR1B	0	RW	**RAM2ERR1B Interrupt Enable**
	Enable/disable the RAM2ERR1B interrupt			
19	RAM1ERR2B	0	RW	**RAM1ERR2B Interrupt Enable**
	Enable/disable the RAM1ERR2B interrupt			
18	RAM1ERR1B	0	RW	**RAM1ERR1B Interrupt Enable**
	Enable/disable the RAM1ERR1B interrupt			
17	RAMERR2B	0	RW	**RAMERR2B Interrupt Enable**
	Enable/disable the RAMERR2B interrupt			
16	RAMERR1B	0	RW	**RAMERR1B Interrupt Enable**
	Enable/disable the RAMERR1B interrupt			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	LVEWRITE	0	RW	**LVEWRITE Interrupt Enable**
	Enable/disable the LVEWRITE interrupt			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	WDATAOV	0	RW	**WDATAOV Interrupt Enable**
	Enable/disable the WDATAOV interrupt			
5	ICACHERR	0	RW	**ICACHERR Interrupt Enable**
	Enable/disable the ICACHERR interrupt			
4	PWRUPF	0	RW	**PWRUPF Interrupt Enable**
	Enable/disable the PWRUPF interrupt			
3	CMOF	0	RW	**CMOF Interrupt Enable**
	Enable/disable the CMOF interrupt			

Bit	Name	Reset	Access	Description
2	CHOF	0	RW	**CHOF Interrupt Enable**
	Enable/disable the CHOF interrupt			
1	WRITE	0	RW	**WRITE Interrupt Enable**
	Enable/disable the WRITE interrupt			
0	ERASE	0	RW	**ERASE Interrupt Enable**
	Enable/disable the ERASE interrupt			

6.5.12 MSC_LOCK - Configuration 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																	0x0000															
Access																	RWH															
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	0x0000	RWH	**Configuration Lock**

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

Mode	Value	Description
Read Operation		
UNLOCKED	0	MSC registers are unlocked
LOCKED	1	MSC registers are locked
Write Operation		
LOCK	0	Lock MSC registers
UNLOCK	0x1B71	Unlock MSC registers

6.5.13 MSC_CACHECMD - Flash Cache Command 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																														0	0	0
Access																														W1	W1	W1
Name																														STOPPC	STARTPC	INVCACHE

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	0	W1	**Stop Performance Counters**
	Use this commant bit to stop the performance counters.			
1	STARTPC	0	W1	**Start Performance Counters**
	Use this command bit to start the performance counters. The performance counters always start counting from 0.			
0	INVCACHE	0	W1	**Invalidate Instruction Cache**
	Use this register to invalidate the instruction cache.			

6.5.14 MSC_CACHEHITS - Cache Hits Performance Counter

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													0x00000																			
Access													R																			
Name													CACHEHITS																			

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	CACHEHITS	0x00000	R	**Cache Hits Since Last Performance Counter Start Command**
	Use to measure cache performance for a particular code section.			

6.5.15 MSC_CACHEMISSES - Cache Misses Performance Counter

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													0x00000																			
Access													R																			
Name													CACHEMISSES																			

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	CACHEMISSES	0x00000	R	**Cache Misses Since Last Performance Counter Start Command** Use to measure cache performance for a particular code section.

6.5.16 MSC_MASSLOCK - Mass Erase Lock 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																	0x0001															
Access																	RWH															
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	0x0001	RWH	**Mass Erase Lock**

Write any other value than the unlock code to lock access the the ERASEMAINn commands. Write the unlock code 631A to enable access. When reading the register, bit 0 is set when the lock is enabled. Locked by default.

Mode	Value	Description
Read Operation		
UNLOCKED	0	Mass erase unlocked
LOCKED	1	Mass erase locked
Write Operation		
LOCK	0	Lock mass erase
UNLOCK	0x631A	Unlock mass erase

6.5.17 MSC_STARTUP - Startup Control

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		0x1				0	1	1			0x001												0x04D									
Access		RW				RW	RW	RW			RW												RW									
Name		STWS				STWSAEN	STWSEN	ASTWAIT			STDLY1												STDLY0									

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	STWS	0x1	RW	**Startup Waitstates**
	Active wait for flash startup startup after SDLY0.			
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	STWSAEN	0	RW	**Startup Waitstates Always Enable**
	Use the number of waitstates given by STWS during startup always.			
25	STWSEN	1	RW	**Startup Waitstates Enable**
	Use the number of waitstates given by STWS during startup. During the optional STDLY1 timeout.			
24	ASTWAIT	1	RW	**Active Startup Wait**
	Active wait for flash startup startup after SDLY0.			
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:12	STDLY1	0x001	RW	**Startup Delay 0**
	Number of cycles with startup waitstates, and also the maximum number of cycles startup sampling will be attempted before starting up system. Note that the reset value of this field may differ from the value shown in this description. The reset value programmed in the device is the optimal value.			
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:0	STDLY0	0x04D	RW	**Startup Delay 0**
	Number of idle cycles from exiting sleep mode. Note that the reset value of this field may differ from the value shown in this description. The reset value programmed in the device is the optimal value.			

6.5.18 MSC_BANKSWITCHLOCK - Bank Switching Lock 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																									0x0001							
Access																									RWH							
Name																									BANKSWITCHLOCKKEY							

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	BANKSWITCHLOCKKEY	0x0001	RWH	**Bank Switching Lock**

Write any other value than the unlock code to lock access the the BankSwitching commands. Write the unlock code 7C2B to enable access. When reading the register, bit 0 is set when the lock is enabled. Locked by default

Mode	Value	Description
Read Operation		
UNLOCKED	0	Bank Switching unlocked
LOCKED	1	Bank Switching locked
Write Operation		
LOCK	0	Lock Bank Switching
UNLOCK	0x7C2B	Unlock Bank Switching

6.5.19 MSC_CMD - Command 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																															0	0
Access																															W1	W1
Name																															SWITCHINGBANK	PWRUP

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	SWITCHINGBANK	0	W1	**BANK SWITCHING COMMAND**
	Write to this bit to manual switching the flash banks if UCFGLOCK1.BANKSWITCHINGDIS is zero. MSC_STATUS.BANKSWITCHED is the status bit			
0	PWRUP	0	W1	**Flash Power Up Command**
	Write to this bit to power up the Flash. IRQ PWRUPF will be fired when power up sequence completed.			

6.5.20 MSC_BOOTLOADERCTRL - Bootloader Read and Write Enable, Write Once 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																															0	0
Access																															RW	RW
Name																															BLWDIS	BLRDIS

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	BLWDIS	0	RW	**Flash Bootloader Write/Erase Disable**
	Controls write/erase access of the flash bootloader pages. When cleared, write/erase is enabled. When set, write/erase is disabled.			
0	BLRDIS	0	RW	**Flash Bootloader Read Disable**
	Controls read access of the flash bootloader pages. When cleared, read is enabled. When set, read is disabled.			

6.5.21 MSC_AAPUNLOCKCMD - Software Unlock AAP Command Register

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																																0
Access																																W1
Name																																UNLOCKAAP

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	UNLOCKAAP	0	W1	**Software Unlock AAP Command** Write to this bit to unlock AAP. This is only possible when bit 31 of the AAP Lock Word (ALW) in flash is set to 1. If bit 31 of the ALW has been cleared to 0, this command has no effect. Register is writable only when MSC_LOCK is unlocked

6.5.22 MSC_CACHECONFIG0 - Cache Configuration Register 0

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																															0x3	
Access																															RW	
Name																															CACHELPLEVEL	

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	CACHELPLEVEL	0x3	RW	**Instruction Cache Low-Power Level** Use this to set the low-power level of the cache. In general, the default setting is best for most applications.

Value	Mode	Description
0	BASE	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.

6.5.23 MSC_RAMCTRL - RAM Control Enable 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														0	0							0	0							0	0	
Access														RW	RW							RW	RW							RW	RW	
Name														RAM2PREFETCHEN	RAM2WSEN							RAM1PREFETCHEN	RAM1WSEN						RAMPREFETCHEN	RAMWSEN	

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	RAM2PREFETCHEN	0	RW	**RAM2 Prefetch Enable**
	RAM2 Prefetch Enable. Setting this bit will enable prefetch access on RAM2 data.			
17	RAM2WSEN	0	RW	**RAM2 WAIT STATE Enable**
	RAM2 WAIT STATE Enable. Set when RAM2 access clock is more than 38 MHz (at 1.2V), setting this bit will add Wait state.			
16:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	RAM1PREFETCHEN	0	RW	**RAM1 Prefetch Enable**
	RAM1 Prefetch Enable. Setting this bit will enable prefetch access on RAM1 data.			
9	RAM1WSEN	0	RW	**RAM1 WAIT STATE Enable**
	RAM1 WAIT STATE Enable. Set when RAM1 access clock is more than 38 MHz (at 1.2V), setting this bit will add Wait state.			
8:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	RAMPREFETCHEN	0	RW	**RAM Prefetch Enable**
	RAM Prefetch Enable. Setting this bit will enable prefetch access on RAM data.			
1	RAMWSEN	0	RW	**RAM WAIT STATE Enable**
	RAM WAIT STATE Enable. Set when RAM access clock is more than 38 MHz (at 1.2V), setting this bit will add Wait state.			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

6.5.24 MSC_ECCCTRL - RAM ECC Control 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																											0	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											RAM2ECCCHKEN	RAM2ECCEWEN	RAM1ECCCHKEN	RAM1ECCEWEN	RAMECCCHKEN	RAMECCEWEN

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	RAM2ECCCHKEN	0	RW	**RAM2 ECC Check Enable**
	RAM2 ECC Check Enable.			
4	RAM2ECCEWEN	0	RW	**RAM2 ECC Write Enable**
	RAM2 ECC Write Enable, when set ECC is enabled			
3	RAM1ECCCHKEN	0	RW	**RAM1 ECC Check Enable**
	RAM1 ECC Check Enable.			
2	RAM1ECCEWEN	0	RW	**RAM1 ECC Write Enable**
	RAM1 ECC Write Enable, when set ECC is enabled			
1	RAMECCCHKEN	0	RW	**RAM ECC Check Enable**
	RAM ECC Check Enable.			
0	RAMECCEWEN	0	RW	**RAM ECC Write Enable**
	RAM ECC Write Enable, when set ECC is enabled			

6.5.25 MSC_RAMECCADDR - RAM ECC Error Address Register

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	0x00000000																															
Access	R																															
Name	RAMECCADDR																															

Bit	Name	Reset	Access	Description
31:0	RAMECCADDR	0x00000000	R	**RAM ECC Error Address**
	Indicates Address of RAM at which ECC error occured			

6.5.26 MSC_RAM1ECCADDR - RAM1 ECC Error Address Register

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	0x00000000																															
Access	R																															
Name	RAM1ECCADDR																															

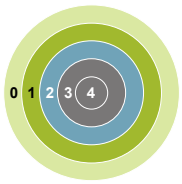
Bit	Name	Reset	Access	Description
31:0	RAM1ECCADDR	0x00000000	R	**RAM1 ECC Error Address**
	Indicates Address of RAM1 at which ECC error occured			

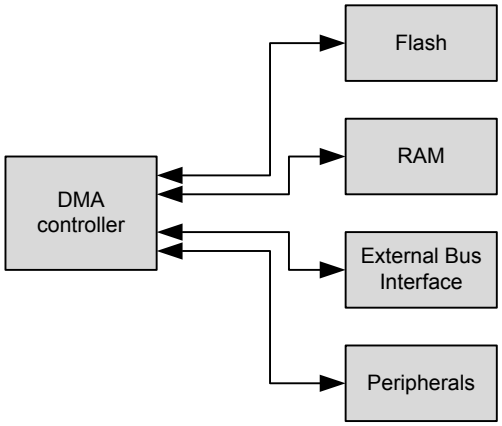
6.5.27 MSC_RAM2ECCADDR - RAM2 ECC Error Address Register

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	0x00000000																															
Access	R																															
Name	RAM2ECCADDR																															

Bit	Name	Reset	Access	Description
31:0	RAM2ECCADDR	0x00000000	R	**RAM2 ECC Error Address**
	Indicates Address of RAM2 at which ECC error occured			

7. LDMA - Linked DMA Controller





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. For example the LEUART can provide full UART communication in EM2 DeepSleep, consuming only a few µA by using the LDMA to move data between the LEUART and RAM.

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.

7.1 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 LEUART to memory or memory to LEUART. Each of the DMA channels on the EFM32 can be connected to any of the EFM32 peripherals.

7.1.1 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

7.2 Block Diagram

An overview of the LDMA and the modules it interacts with is shown in Figure 7.1 LDMA Block Diagram on page 207.

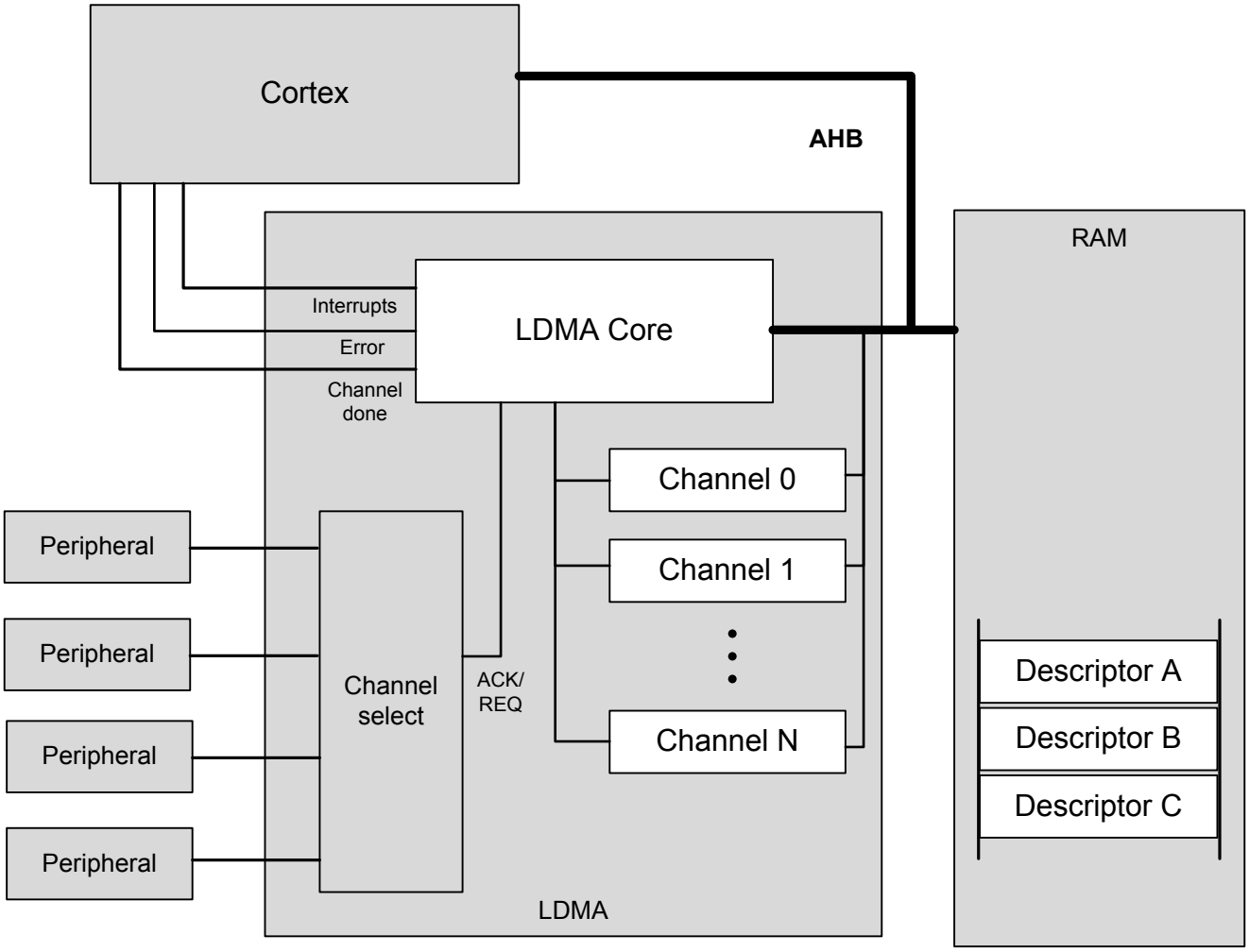


Figure 7.1. LDMA Block Diagram

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.

7.3 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 12 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 7.3.2 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 7.3.7 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.

7.3.1 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.

7.3.1.1 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.

7.3.1.2 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.

SRCINC or DSTINC may also be set to NONE which will cause the LDMA to read or write the same location for every DMA transfer. This is useful for accessing peripheral FIFO or data registers.

7.3.1.3 Block Size

The block size defines the amount of data transferred in one arbitration. It consists of one or more DMA transfers. See 7.3.6.1 Arbitration Priority for more details.

7.3.1.4 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

7.3.1.5 Descriptor List

A descriptor list consists of one or more descriptors which are executed in 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 7.3.2.2 Loop Counter).
- Sync descriptor: Handle synchronization of the list with other entities (see 7.3.7.2 SYNC Descriptor Structure).
- WRI descriptor: Writes a value to a location in memory (see 7.3.7.3 WRI Descriptor Structure).

7.3.1.6 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 desriptor 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.

7.3.1.7 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 7.4.6 2D Copy for an common use of relative addressing.

7.3.1.8 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.

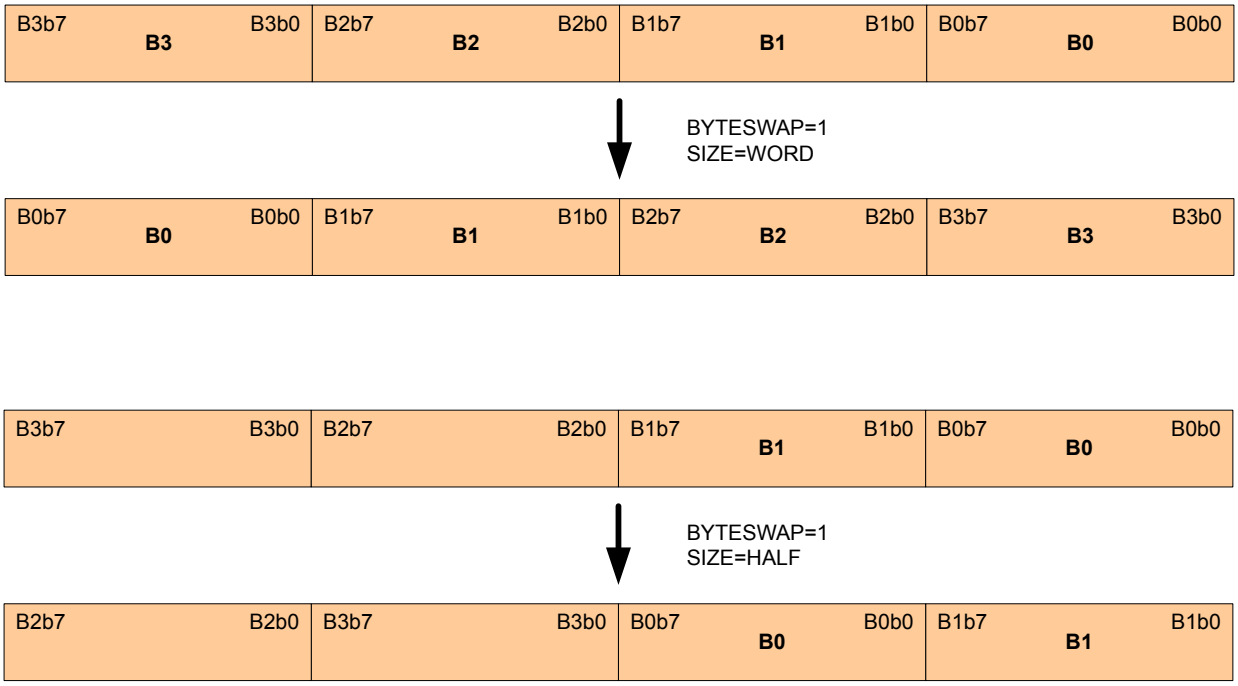


Figure 7.2. Word and Half-Word Endian Byte Swap Examples

7.3.1.9 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.

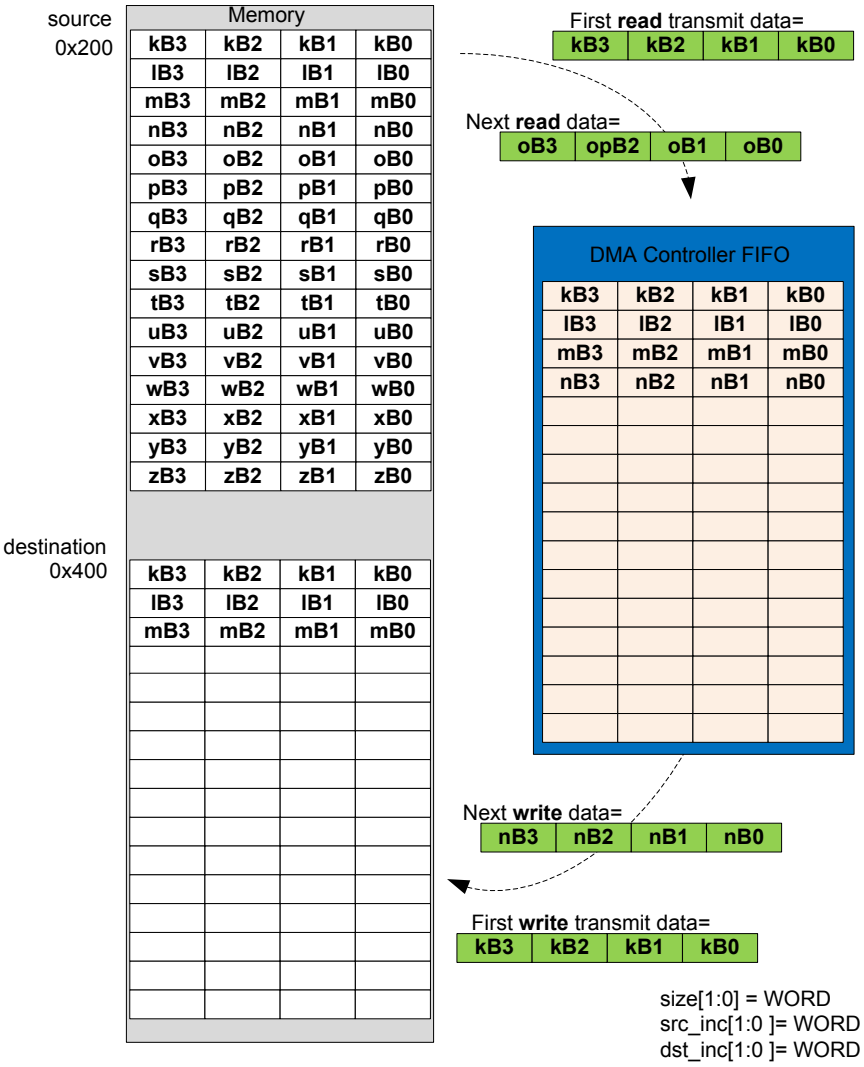


Figure 7.3. Memory-to-Memory Transfer WORD Size Example

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.

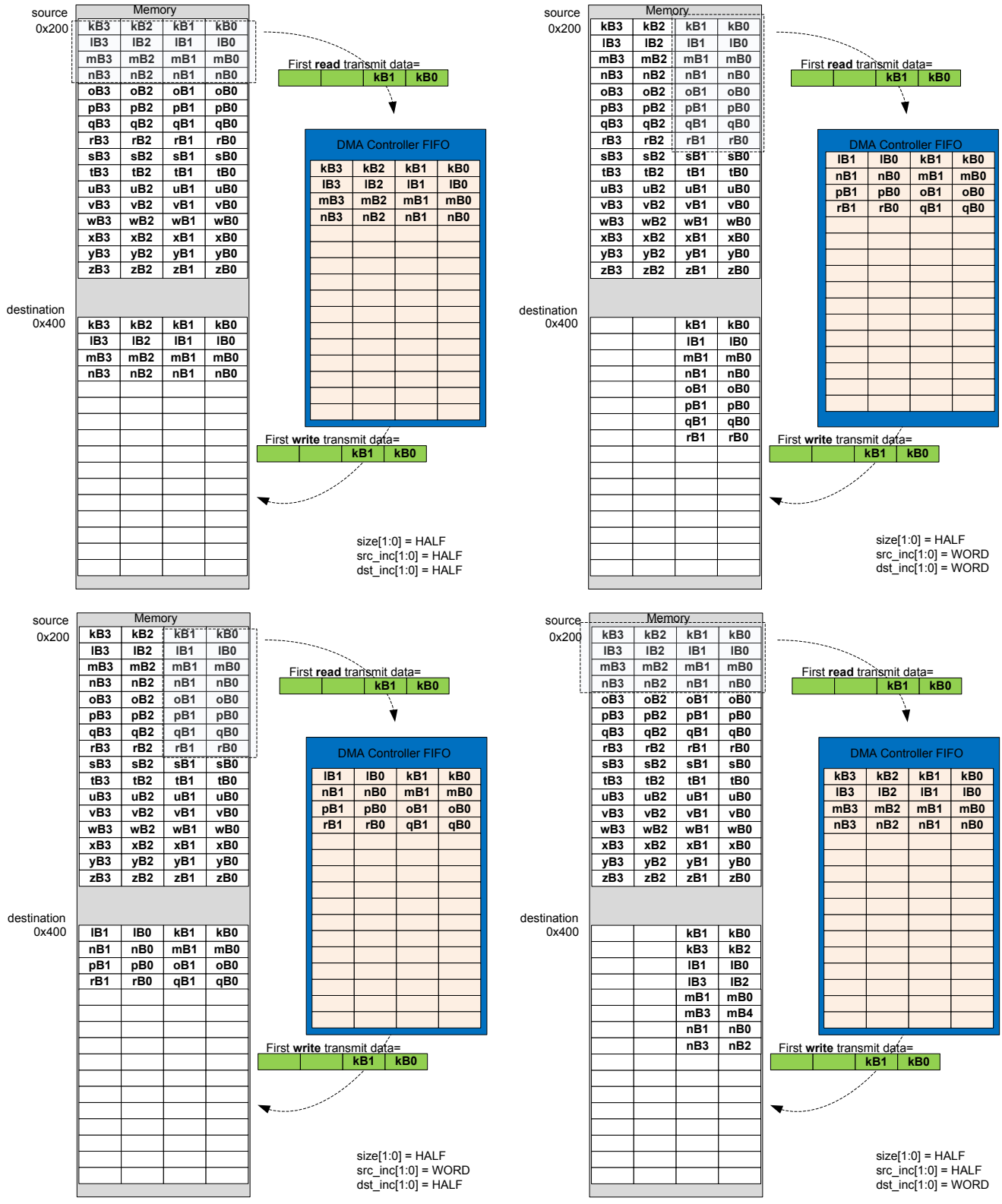


Figure 7.4. Memory-to-Memory Transfer HALF Size Examples

Fields SRCINCSIGN and DSTINCSIGN allow for address decrement. These can be used to mirror an image, for example, in the pixel copy application.

7.3.2 Channel Configuration

Each DMA channel has associated configuration and loop counter registers for controlling direction of address increment , arbitration slots, and descriptor looping.

7.3.2.1 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.

7.3.2.2 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.

7.3.3 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.

7.3.4 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 7.3.3 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.

7.3.4.1 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.

7.3.5 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.

7.3.6 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.

7.3.6.1 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 7.1. 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.

7.3.6.2 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 7.2 AHB Bus Transfer Arbitration Interval on page 216 lists the arbitration rates.

Table 7.2. 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 7.3.7.1 XFER Descriptor Structure for more information about the location of the BLOCKSIZE bits in the data structure.

7.3.7 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**

7.3.7.1 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. Refer to the register descriptions for additional information.

For specifying XFER structure type, set STRUCTTYPE to 0. 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

7.3.7.2 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.

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																	SYNCCLR								SYNCSET						
DST																	MATCHEN								MATCHVAL						
LINK	LINKADDR																													LINK	LINKMODE	

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).

7.3.7.3 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	IMMVAL																															
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.

7.3.8 Interaction With the EMU

The LDMA interacts with the Energy Management Unit (EMU) to allow transfers from a low energy peripheral while in EM2 DeepSleep. For example, when using the LEUART in EM2 DeepSleep the EMU can wake up the LDMA sufficiently long to allow data transfers to occur. See section "DMA Support" in the LEUART documentation.

Similarly, when using the ADC in EM2 DeepSleep or EM3 Stop the EMU can wake up the LDMA as needed to allow data transfers to occur.

Table 7.3 List of Peripherals Capable of Waking Up LDMA in EM2 DeepSleep or EM3 Stop on page 220 shows complete list of peripherals that are capable of waking up LDMA via EMU in EM2 DeepSleep or EM3 Stop

Table 7.3. List of Peripherals Capable of Waking Up LDMA in EM2 DeepSleep or EM3 Stop

Peripheral
ADC0
ADC1
CSEN
IDAC0
LESENSE
LEUART0
LEUART1

7.3.9 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 Cortex-M4 core. They are combined into one interrupt vector, DMA_INT. If the interrupt for the DMA is enabled in the ARM Cortex-M4 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 immediatly 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.

7.3.10 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.

7.4 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.

7.4.1 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.

7.4.2 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 descritpors 2-4 will cause them to begin transfering 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 7.5 Descriptor Linked List on page 222. 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 7.5 Descriptor Linked List on page 222.

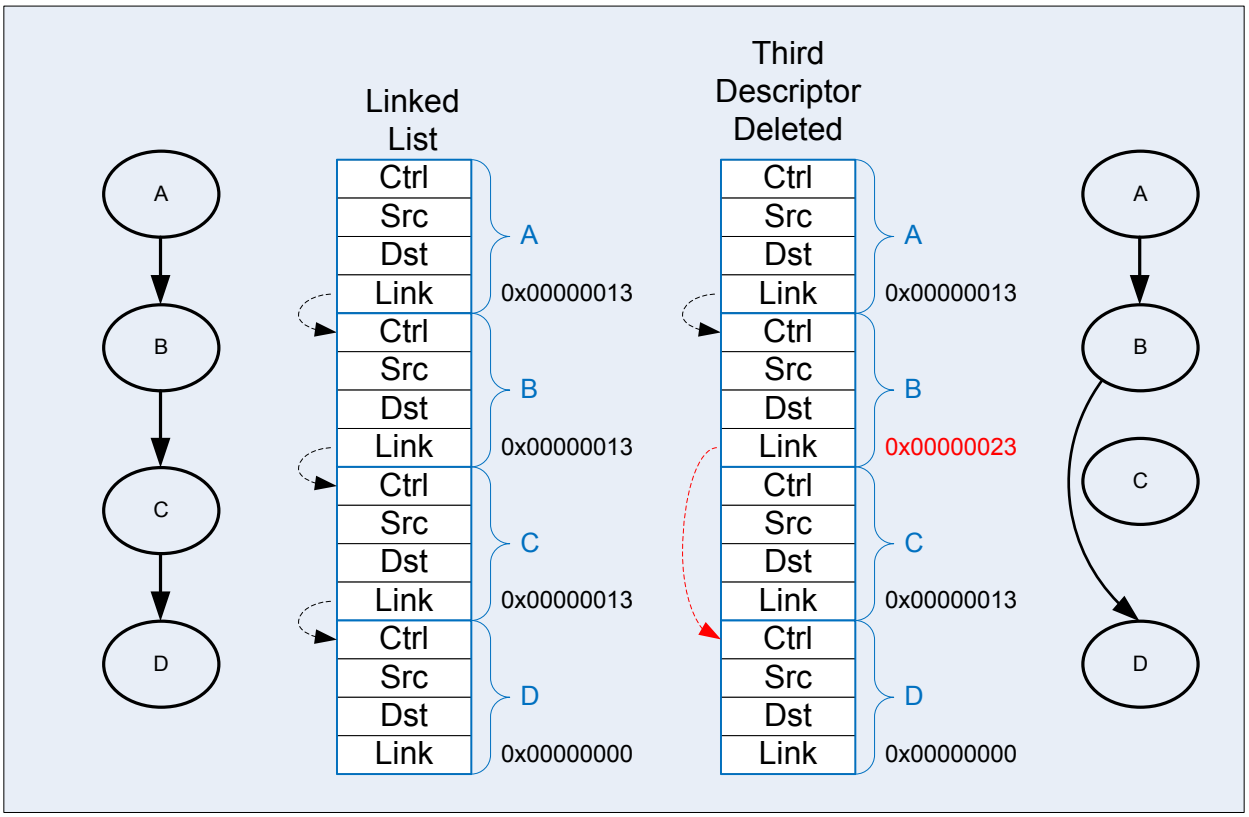


Figure 7.5. Descriptor Linked List

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

Alternativley, 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.

7.4.3 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 transfered.

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 LINKLink 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 7.4.2 Descriptor Linked List

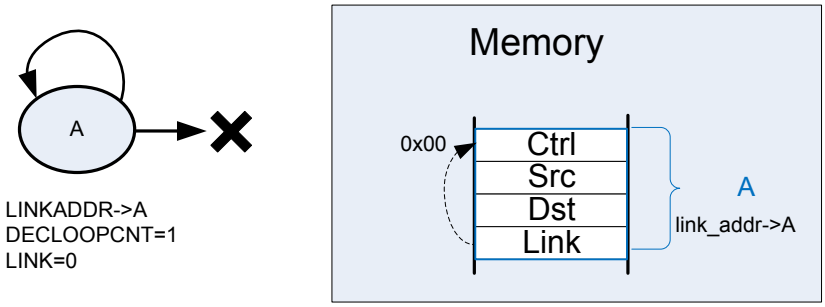


Figure 7.6. Single Descriptor Looped Transfer

Note that the looping descriptor must be stored in memory, because it must load itself from the link address in memory on each iteration.

7.4.4 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 7.7 Descriptor List With Looping on page 225. 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.

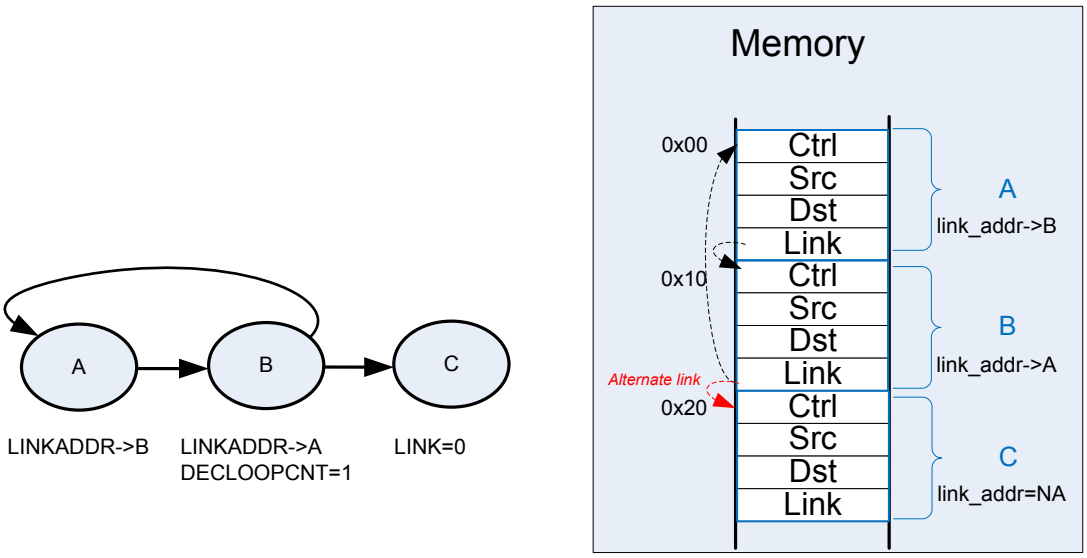


Figure 7.7. Descriptor List With Looping

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

7.4.5 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])
```

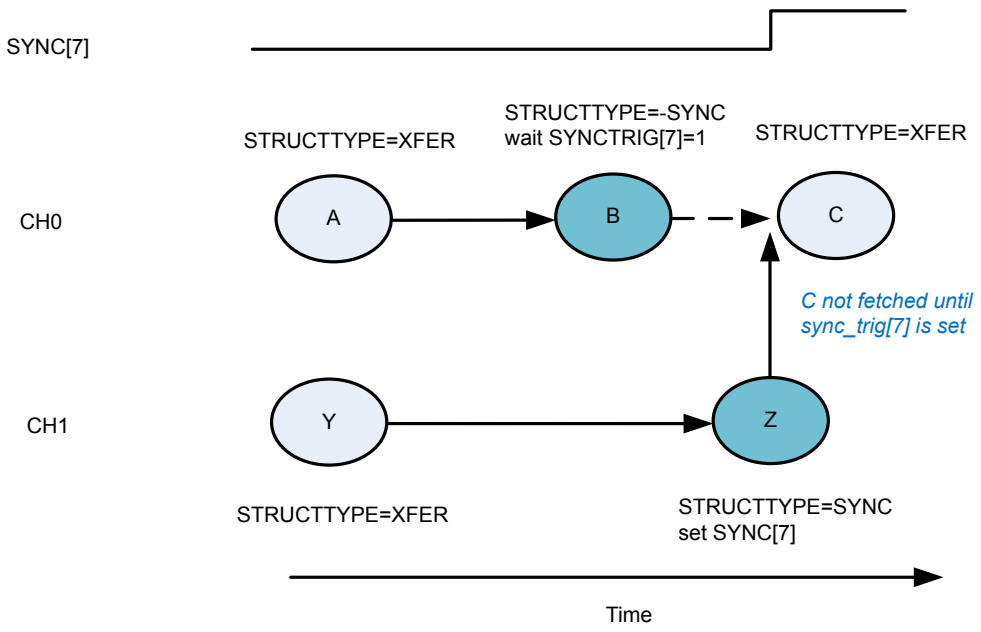


Figure 7.8. Simple Intra-channel Synchronization Example

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.

7.4.6 2D Copy

The LDMA can easily perform a 2D copy using a descriptor list with looping. This set up is visualized in Figure 7.9 2D Copy on page 228.

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.

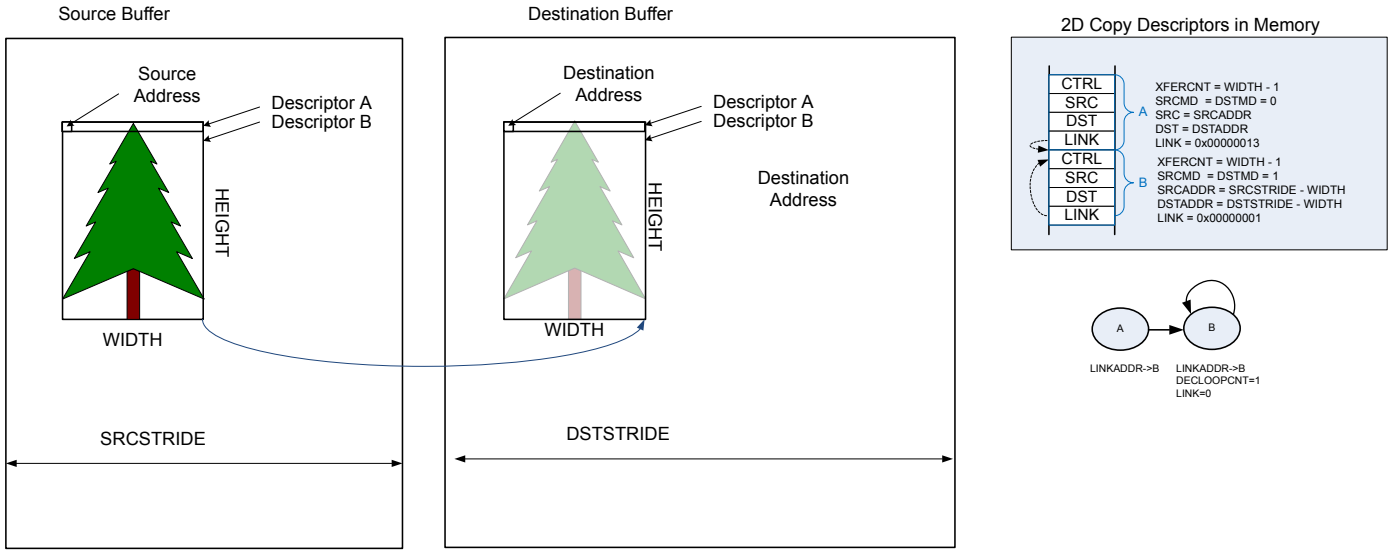


Figure 7.9. 2D Copy

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.

7.4.7 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 7.10 Infinite Ping-Pong Example on page 230. 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.

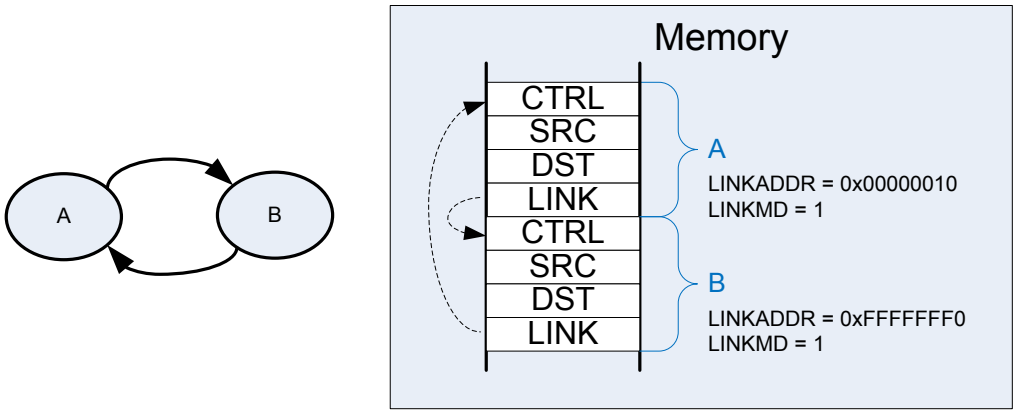


Figure 7.10. Infinite Ping-Pong Example

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)

7.4.8 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 7.4.2 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.

7.5 Register Map

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

Offset	Name	Type	Description
0x000	LDMA_CTRL	RW	DMA Control Register
0x004	LDMA_STATUS	R	DMA Status Register
0x008	LDMA_SYNC	RWH	DMA Synchronization Trigger Register (Single-Cycle RMW)
0x020	LDMA_CHEN	RWH	DMA Channel Enable Register (Single-Cycle RMW)
0x024	LDMA_CHBUSY	R	DMA Channel Busy Register
0x028	LDMA_CHDONE	RWH	DMA Channel Linking Done Register (Single-Cycle RMW)
0x02C	LDMA_DBGHALT	RW	DMA Channel Debug Halt Register
0x030	LDMA_SWREQ	W1	DMA Channel Software Transfer Request Register
0x034	LDMA_REQDIS	RW	DMA Channel Request Disable Register
0x038	LDMA_REQPEND	R	DMA Channel Requests Pending Register
0x03C	LDMA_LINKLOAD	W1	DMA Channel Link Load Register
0x040	LDMA_REQCLEAR	W1	DMA Channel Request Clear Register
0x060	LDMA_IF	R	Interrupt Flag Register
0x064	LDMA_IFS	W1	Interrupt Flag Set Register
0x068	LDMA_IFC	(R)W1	Interrupt Flag Clear Register
0x06C	LDMA_IEN	RW	Interrupt Enable Register
0x080	LDMA_CH0_REQSEL	RW	Channel Peripheral Request Select Register
0x084	LDMA_CH0_CFG	RW	Channel Configuration Register
0x088	LDMA_CH0_LOOP	RWH	Channel Loop Counter Register
0x08C	LDMA_CH0_CTRL	RWH	Channel Descriptor Control Word Register
0x090	LDMA_CH0_SRC	RWH	Channel Descriptor Source Data Address Register
0x094	LDMA_CH0_DST	RWH	Channel Descriptor Destination Data Address Register
0x098	LDMA_CH0_LINK	RWH	Channel Descriptor Link Structure Address Register
...	LDMA_CHx_REQSEL	RW	Channel Peripheral Request Select Register
...	LDMA_CHx_CFG	RW	Channel Configuration Register
...	LDMA_CHx_LOOP	RWH	Channel Loop Counter Register
...	LDMA_CHx_CTRL	RWH	Channel Descriptor Control Word Register
...	LDMA_CHx_SRC	RWH	Channel Descriptor Source Data Address Register
...	LDMA_CHx_DST	RWH	Channel Descriptor Destination Data Address Register
...	LDMA_CHx_LINK	RWH	Channel Descriptor Link Structure Address Register
0x290	LDMA_CH11_REQSEL	RW	Channel Peripheral Request Select Register
0x294	LDMA_CH11_CFG	RW	Channel Configuration Register
0x298	LDMA_CH11_LOOP	RWH	Channel Loop Counter Register
0x29C	LDMA_CH11_CTRL	RWH	Channel Descriptor Control Word Register
0x2A0	LDMA_CH11_SRC	RWH	Channel Descriptor Source Data Address Register

Offset	Name	Type	Description
0x2A4	LDMA_CH11_DST	RWH	Channel Descriptor Destination Data Address Register
0x2A8	LDMA_CH11_LINK	RWH	Channel Descriptor Link Structure Address Register

7.6 Register Description

7.6.1 LDMA_CTRL - DMA 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					0xB												0x00								0x00							
Access					RW												RW								RW							
Name					NUMFIXED												SYNCPRSCLREN								SYNCPRSSETEN							

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	NUMFIXED	0xB	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.
23:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	SYNCPRSCLREN	0x00	RW	**Synchronization PRS Clear Enable** Setting a bit in this field will enable the corresponding PRS input to clear the respective bit in the SYNCTRIG field of the LDMA_SYNC register. Refer to the PRS section for a list of the PRS inputs.
7:0	SYNCPRSSETEN	0x00	RW	**Synchronization PRS Set Enable** Setting a bit in this field will enable the corresponding PRS input to set the respective bit in the SYNCTRIG field of the LDMA_SYNC register. Refer to the PRS section for a list of the PRS inputs.

7.6.2 LDMA_STATUS - DMA Status 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				0x0C								0x10									0x0					0x0					0	0
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	0x0C	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.)
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	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	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:3	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	0	R	**Any DMA Channel Request Pending** The value of this bit will be TRUE (1) if any requests are pending
0	ANYBUSY	0	R	**Any DMA Channel Busy** The value of this bit will be TRUE (1) if one or more DMA channels are actively transferring data

7.6.3 LDMA_SYNC - DMA Synchronization Trigger Register (Single-Cycle RMW)

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																									0x00							
Access																									RWH							
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	0x00	RWH	**Synchronization Trigger**
	The SYNC trigger field allows a transfer to pause until a specified trigger bit is set or cleared. The SYNC trigger bits may be set and cleared by a SYNC descriptor, PRS signal, or software. Note: software requires to use single-cycle read-modify-write, detailed in 4.2.3 Peripheral Bit Set and Clear			

7.6.4 LDMA_CHEN - DMA Channel Enable Register (Single-Cycle RMW)

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																										0x000						
Access																										RWH						
Name																										CHEN						

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	CHEN	0x000	RWH	**Channel Enables**
	Setting one of these bits will enable the respective DMA channel. If cleared while a transfer is in progress, the current transfer block will complete. The remaining blocks will pause until resumed later by setting this bit again. Note: software requires to use single-cycle read-modify-write, detailed in 4.2.3 Peripheral Bit Set and Clear			

7.6.5 LDMA_CHBUSY - DMA Channel 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																					0x000											
Access																					R											
Name																					BUSY											

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	BUSY	0x000	R	**Channels Busy**
	The bits of this field read 1 when the corresponding channel is busy.			

7.6.6 LDMA_CHDONE - DMA Channel Linking Done Register (Single-Cycle RMW)

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																					0x000											
Access																					RWH											
Name																					CHDONE											

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	CHDONE	0x000	RWH	**DMA Channel Linking or Done**
	Each DMA channel sets the corresponding bit in this register when the entire transfer is done. The interrupt service routine should clear these bits. Enabling a DMA channel will also clear the corresponding LINKDONE bit. Note: software requires to use single-cycle read-modify-write, detailed in 4.2.3 Peripheral Bit Set and Clear			

7.6.7 LDMA_DBGHALT - DMA Channel Debug Halt 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																											0x000					
Access																											RW					
Name																											DBGHALT					

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	DBGHALT	0x000	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.			

7.6.8 LDMA_SWREQ - DMA Channel Software Transfer Request 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																											0x000					
Access																											W1					
Name																											SWREQ					

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	SWREQ	0x000	W1	**Software Transfer Requests**
	Setting one of these bits will trigger a DMA transfer for the corresponding channel. Writing zeros has no effect.			

7.6.9 LDMA_REQDIS - DMA Channel Request Disable 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																										0x000						
Access																										RW						
Name																										REQDIS						

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	REQDIS	0x000	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.			

7.6.10 LDMA_REQPEND - DMA Channel Requests Pending 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																										0x000						
Access																										R						
Name																										REQPEND						

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	REQPEND	0x000	R	**DMA Requests Pending**
	When a DMA channel has a pending peripheral request the corresponding REQPEND bit will read 1.			

7.6.11 LDMA_LINKLOAD - DMA Channel Link Load 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																											0x000					
Access																											W1					
Name																											LINKLOAD					

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	LINKLOAD	0x000	W1	**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.			

7.6.12 LDMA_REQCLEAR - DMA Channel Request Clear 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																											0x000					
Access																											W1					
Name																											REQCLEAR					

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	REQCLEAR	0x000	W1	**DMA Request Clear**
	Setting one of these bits will clear any internally registered transfer requests for the corresponding channel.			

7.6.13 LDMA_IF - Interrupt Flag 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	0																				0x000											
Access	R																				R											
Name	ERROR																				DONE											

Bit	Name	Reset	Access	Description
31	ERROR	0	R	**Transfer Error Interrupt Flag**
	The ERRORIF flag is set when a read or write error occurs. The CHERROR field in the LDMA_STATUS register reflects the number of the channel which had the last error.			
30:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	DONE	0x000	R	**DMA Structure Operation Done Interrupt Flag**
	When a channel completes a transfer or sync operation, the corresponding DONE bit is set in the LDMA_IF register.			

7.6.14 LDMA_IFS - Interrupt Flag Set 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	0																				0x000											
Access	W1																				W1											
Name	ERROR																				DONE											

Bit	Name	Reset	Access	Description
31	ERROR	0	W1	**Set ERROR Interrupt Flag**
	Write 1 to set the ERROR interrupt flag			
30:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	DONE	0x000	W1	**Set DONE Interrupt Flag**
	Write 1 to set the DONE interrupt flag			

7.6.15 LDMA_IFC - Interrupt Flag Clear 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	0																				0x000											
Access	(R)W1																				(R)W1											
Name	ERROR																				DONE											

Bit	Name	Reset	Access	Description
31	ERROR	0	(R)W1	**Clear ERROR Interrupt Flag**
	Write 1 to clear the ERROR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
30:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	DONE	0x000	(R)W1	**Clear DONE Interrupt Flag**
	Write 1 to clear the DONE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

7.6.16 LDMA_IEN - Interrupt Enable Register

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	0																				0x000											
Access	RW																				RW											
Name	ERROR																				DONE											

Bit	Name	Reset	Access	Description
31	ERROR	0	RW	**ERROR Interrupt Enable**
	Enable/disable the ERROR interrupt			
30:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	DONE	0x000	RW	**DONE Interrupt Enable**
	Enable/disable the DONE interrupt			

7.6.17 LDMA_CHx_REQSEL - Channel Peripheral Request Select 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											0x00																		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	0x00	RW	**Source Select**

Select input source to DMA channel.

Value	Mode	Description
0b000000	NONE	No source selected
0b000001	PRS	Peripheral Reflex System
0b001000	ADC0	Analog to Digital Converter 0
0b001001	ADC1	Analog to Digital Converter 0
0b001010	VDAC0	Digital to Analog Converter 0
0b001100	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter 0
0b001101	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter 1
0b001110	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter 2
0b001111	USART3	Universal Synchronous/Asynchronous Receiver/Transmitter 3
0b010000	USART4	Universal Synchronous/Asynchronous Receiver/Transmitter 4
0b010010	UART0	Universal Asynchronous Receiver/Transmitter 0
0b010011	UART1	Universal Asynchronous Receiver/Transmitter 1
0b010100	LEUART0	Low Energy UART 0
0b010101	LEUART1	Low Energy UART 1
0b010110	I2C0	I2C 0
0b010111	I2C1	I2C 1
0b011001	TIMER0	Timer 0
0b011010	TIMER1	Timer 1
0b011011	TIMER2	Timer 2
0b011100	TIMER3	Timer 3
0b100000	WTIMER0	Wide Timer 0
0b100001	WTIMER1	Wide Timer 0

Bit	Name	Reset	Access	Description
	0b110000	MSC		Memory System Controller
	0b110001	CRYPTO0		Advanced Encryption Standard Accelerator
	0b110010	EBI		External Bus Interface
	0b110011	PDM		PDM Interface
	0b111101	CSEN		Capacitive touch sense module
	0b111110	LESENSE		Low Energy Sensor Interface
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.

Value	Mode	Description
SOURCESEL =	*0b000000*	*(NONE)*
0bxxxx	OFF	Channel input selection is turned off
SOURCESEL =	*0b000001*	*(PRS)*
0b0000	PRSREQ0	PRSREQ0
0b0001	PRSREQ1	PRSREQ1
SOURCESEL =	*0b001000*	*(ADC0)*
0b0000	ADC0SINGLE	ADC0SINGLE REQ/SREQ
0b0001	ADC0SCAN	ADC0SCAN REQ/SREQ
SOURCESEL =	*0b001001*	*(ADC1)*
0b0000	ADC1SINGLE	ADC1SINGLE REQ/SREQ
0b0001	ADC1SCAN	ADC1SCAN REQ/SREQ
SOURCESEL =	*0b001010*	*(VDAC0)*
0b0000	VDAC0CH0	VDAC0CH0
0b0001	VDAC0CH1	VDAC0CH1
SOURCESEL =	*0b001100*	*(USART0)*
0b0000	USART0RXDATAV	USART0RXDATAV REQ/SREQ
0b0001	USART0TXBL	USART0TXBL REQ/SREQ
0b0010	USART0TXEMPTY	USART0TXEMPTY
SOURCESEL =	*0b001101*	*(USART1)*
0b0000	USART1RXDATAV	USART1RXDATAV REQ/SREQ
0b0001	USART1TXBL	USART1TXBL REQ/SREQ
0b0010	USART1TXEMPTY	USART1TXEMPTY
0b0011	USART1RXDATAV-RIGHT	USART1RXDATAVRIGHT REQ/SREQ
0b0100	USART1TXBLRIGHT	USART1TXBLRIGHT REQ/SREQ
SOURCESEL =	*0b001110*	*(USART2)*

Bit	Name	Reset	Access	Description
	0b0000	USART2RXDATAV		USART2RXDATAV REQ/SREQ
	0b0001	USART2TXBL		USART2TXBL REQ/SREQ
	0b0010	USART2TXEMPTY		USART2TXEMPTY
	SOURCESEL =	*0b001111*		*(USART3)*
	0b0000	USART3RXDATAV		USART3RXDATAV REQ/SREQ
	0b0001	USART3TXBL		USART3TXBL REQ/SREQ
	0b0010	USART3TXEMPTY		USART3TXEMPTY
	0b0011	USART3RXDATAV-RIGHT		USART3RXDATAVRIGHT REQ/SREQ
	0b0100	USART3TXBLRIGHT		USART3TXBLRIGHT REQ/SREQ
	SOURCESEL =	*0b010000*		*(USART4)*
	0b0000	USART4RXDATAV		USART4RXDATAV REQ/SREQ
	0b0001	USART4TXBL		USART4TXBL REQ/SREQ
	0b0010	USART4TXEMPTY		USART4TXEMPTY
	0b0011	USART4RXDATAV-RIGHT		USART4RXDATAVRIGHT REQ/SREQ
	0b0100	USART4TXBLRIGHT		USART4TXBLRIGHT REQ/SREQ
	SOURCESEL =	*0b010010*		*(UART0)*
	0b0000	UART0RXDATAV		UART0RXDATAV REQ/SREQ
	0b0001	UART0TXBL		UART0TXBL REQ/SREQ
	0b0010	UART0TXEMPTY		UART0TXEMPTY
	SOURCESEL =	*0b010011*		*(UART1)*
	0b0000	UART1RXDATAV		UART1RXDATAV REQ/SREQ
	0b0001	UART1TXBL		UART1TXBL REQ/SREQ
	0b0010	UART1TXEMPTY		UART1TXEMPTY
	SOURCESEL =	*0b010100*		*(LEUART0)*
	0b0000	LEUART0RXDATAV		LEUART0RXDATAV
	0b0001	LEUART0TXBL		LEUART0TXBL
	0b0010	LEUART0TXEMPTY		LEUART0TXEMPTY
	SOURCESEL =	*0b010101*		*(LEUART1)*
	0b0000	LEUART1RXDATAV		LEUART1RXDATAV
	0b0001	LEUART1TXBL		LEUART1TXBL
	0b0010	LEUART1TXEMPTY		LEUART1TXEMPTY
	SOURCESEL =	*0b010110*		*(I2C0)*
	0b0000	I2C0RXDATAV		I2C0RXDATAV REQ/SREQ
	0b0001	I2C0TXBL		I2C0TXBL REQ/SREQ
	SOURCESEL =	*0b010111*		*(I2C1)*
	0b0000	I2C1RXDATAV		I2C1RXDATAV REQ/SREQ

Bit	Name	Reset	Access	Description
	0b0001	I2C1TXBL		I2C1TXBL REQ/SREQ
	SOURCESEL =	*0b011001*		*(TIMER0)*
	0b0000	TIMER0UFOF		TIMER0UFOF
	0b0001	TIMER0CC0		TIMER0CC0
	0b0010	TIMER0CC1		TIMER0CC1
	0b0011	TIMER0CC2		TIMER0CC2
	SOURCESEL =	*0b011010*		*(TIMER1)*
	0b0000	TIMER1UFOF		TIMER1UFOF
	0b0001	TIMER1CC0		TIMER1CC0
	0b0010	TIMER1CC1		TIMER1CC1
	0b0011	TIMER1CC2		TIMER1CC2
	0b0100	TIMER1CC3		TIMER1CC3
	SOURCESEL =	*0b011011*		*(TIMER2)*
	0b0000	TIMER2UFOF		TIMER2UFOF
	0b0001	TIMER2CC0		TIMER2CC0
	0b0010	TIMER2CC1		TIMER2CC1
	0b0011	TIMER2CC2		TIMER2CC2
	SOURCESEL =	*0b011100*		*(TIMER3)*
	0b0000	TIMER3UFOF		TIMER3UFOF
	0b0001	TIMER3CC0		TIMER3CC0
	0b0010	TIMER3CC1		TIMER3CC1
	0b0011	TIMER3CC2		TIMER3CC2
	SOURCESEL =	*0b100000*		*(WTIMER0)*
	0b0000	WTIMER0UFOF		WTIMER0UFOF
	0b0001	WTIMER0CC0		WTIMER0CC0
	0b0010	WTIMER0CC1		WTIMER0CC1
	0b0011	WTIMER0CC2		WTIMER0CC2
	SOURCESEL =	*0b100001*		*(WTIMER1)*
	0b0000	WTIMER1UFOF		WTIMER1UFOF
	0b0001	WTIMER1CC0		WTIMER1CC0
	0b0010	WTIMER1CC1		WTIMER1CC1
	0b0011	WTIMER1CC2		WTIMER1CC2
	0b0100	WTIMER1CC3		WTIMER1CC3
	SOURCESEL =	*0b110000*		*(MSC)*
	0b0000	MSCWDATA		MSCWDATA REQ/SREQ
	SOURCESEL =	*0b110001*		*(CRYPTO0)*
	0b0000	CRYPTO0DATA0WR		CRYPTO0DATA0WR

Bit	Name	Reset	Access	Description
	0b0001	CRYPTO0DATA0XWR		CRYPTO0DATA0XWR
	0b0010	CRYPTO0DATA0RD		CRYPTO0DATA0RD
	0b0011	CRYPTO0DATA1WR		CRYPTO0DATA1WR
	0b0100	CRYPTO0DATA1RD		CRYPTO0DATA1RD
	SOURCESEL =	*0b110010*		*(EBI)*
	0b0000	EBIPXL0EMPTY		EBIPXL0EMPTY
	0b0001	EBIPXL1EMPTY		EBIPXL1EMPTY
	0b0010	EBIPXLFULL		EBIPXLFULL
	0b0011	EBIDDEMPTY		EBIDDEMPTY
	0b0100	EBIVSYNC		EBIVSYNC
	0b0101	EBIHSYNC		EBIHSYNC
	SOURCESEL =	*0b110011*		*(PDM)*
	0b0000	PDMRXDATAV		PDMRXDATAV REQ/SREQ
	SOURCESEL =	*0b111101*		*(CSEN)*
	0b0000	CSENDATA		CSENDATA
	0b0001	CSENBSLN		CSENBSLN
	SOURCESEL =	*0b111110*		*(LESENSE)*
	0b0000	LESENSEBUFDATAV		LESENSEBUFDATAV REQ/SREQ

7.6.18 LDMA_CHx_CFG - Channel Configuration 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											0	0			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	0	RW	**Destination Address Increment Sign**

Value	Mode	Description
0	POSITIVE	Increment destination address
1	NEGATIVE	Decrement destination address

Bit	Name	Reset	Access	Description
20	SRCINCSIGN	0	RW	**Source Address Increment Sign**

Value	Mode	Description
0	POSITIVE	Increment source address
1	NEGATIVE	Decrement source address

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	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

Bit	Name	Reset	Access	Description
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

7.6.19 LDMA_CHx_LOOP - Channel Loop Counter Register

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																									0x00							
Access																									RWH							
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	0x00	RWH	**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.

7.6.20 LDMA_CHx_CTRL - Channel Descriptor Control Word Register

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	0	0	0x0		0x0		0x0		0	0	0	0	0x0				0	0x000											0		0x0	
Access	R	R	RWH		RWH		RWH		RWH	RWH	RWH	RWH	RWH				RWH	RWH											W1		R	
Name	DSTMODE	SRCMODE	DSTINC		SIZE		SRCINC		IGNORESREQ	DECLOOPCNT	REQMODE	DONEIFSEN	BLOCKSIZE				BYTESWAP	XFERCNT											STRUCTREQ		STRUCTTYPE	

Bit	Name	Reset	Access	Description
31	DSTMODE	0	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	0	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	RWH	**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	RWH	**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	RWH	**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	0	RWH	**Ignore Sreq**

The channel arbiter will ignore single requests (SREQ) and only respond to multiple requests (REQ) when this bit is set.

22	DECLOOPCNT	0	RWH	**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	0	RWH	**DMA Request Transfer Mode Select**

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	DONEIFSEN	0	RWH	**DMA Operation Done Interrupt Flag Set Enable**

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	RWH	**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
4	UNIT6	Six unit transfers per arbitration

Bit	Name	Reset	Access	Description
	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	0	RWH	**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	0x000	RWH	**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	0	W1	**Structure DMA Transfer Request**
	When a linked descriptor is loaded with this bit set, it will immediately trigger a transfer.			
2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	STRUCTTYPE	0x0	R	**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.

7.6.21 LDMA_CHx_SRC - Channel Descriptor Source Data Address 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	0x00000000																															
Access	RWH																															
Name	SRCADDR																															

Bit	Name	Reset	Access	Description
31:0	SRCADDR	0x00000000	RWH	**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.			

7.6.22 LDMA_CHx_DST - Channel Descriptor Destination Data Address Register

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	0x00000000																															
Access	RWH																															
Name	DSTADDR																															

Bit	Name	Reset	Access	Description
31:0	DSTADDR	0x00000000	RWH	**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.			

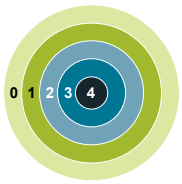
7.6.23 LDMA_CHx_LINK - Channel Descriptor Link Structure Address Register

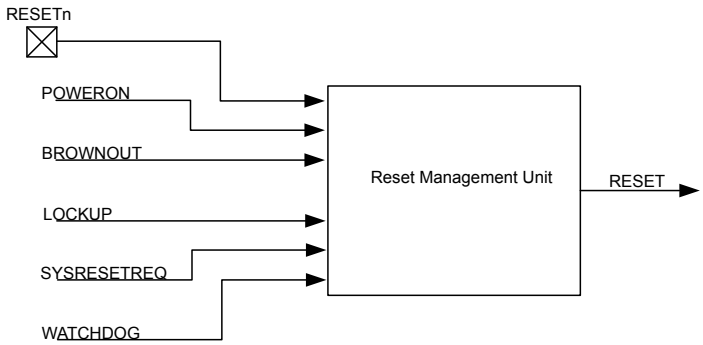
Offset	Bit Position																													
0x098	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																														0	0
Access	RWH																														RWH	R
Name	LINKADDR																														LINK	LINKMODE

Bit	Name	Reset	Access	Description
31:2	LINKADDR	0x00000000	RWH	**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	0	RWH	**Link Next Structure**
	After completing the initial transfer, if this bit is 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	0	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.

8. RMU - Reset Management Unit





Quick Facts

What?

The RMU ensures correct reset operation. It is responsible for connecting the different reset sources to the reset lines of the EFM32 Giant Gecko 12.

Why?

A correct reset sequence is needed to ensure safe and synchronous startup of the EFM32 Giant Gecko 12. In the case of error situations such as power supply glitches or software crash, the RMU provides proper reset and startup of the EFM32 Giant Gecko 12.

How?

The Power-on Reset and Brown-out Detector of the EFM32 Giant Gecko 12 provides power line monitoring with exceptionally low power consumption. The cause of the reset may be read from a register, thus providing software with information about the cause of the reset.

8.1 Introduction

The RMU is responsible for handling the reset functionality of the EFM32 Giant Gecko 12.

8.2 Features

- Reset sources
  - Power-on Reset (POR)
  - Brown-out Detection (BOD) on the following power domains:
    - Analog Unregulated Power Domain AVDD
    - Digital Unregulated Power Domain DVDD
    - Regulated Digital Domain DECOUPLE (DEC)
  - RESETn pin reset
  - Watchdog reset
  - Software triggered reset (SYSRESETREQ)
  - Core LOCKUP condition
- EM4 Hibernate/Shutoff Detection
- EM4 Hibernate/Shutoff wakeup reset from GPIO pin
- Configurable reset levels
- A software readable register indicates the cause of the last reset

8.3 Functional Description

The RMU monitors each of the reset sources of the EFM32 Giant Gecko 12. If one or more reset sources go active, the RMU applies reset to the EFM32 Giant Gecko 12. When the reset sources go inactive the EFM32 Giant Gecko 12 starts up. At startup the EFM32 Giant Gecko 12 loads the stack pointer and program entry point from memory, and starts execution. Figure 8.1 RMU Reset Input Sources and Connections on page 255 shows an overview of the reset system on EFM32 Giant Gecko 12.

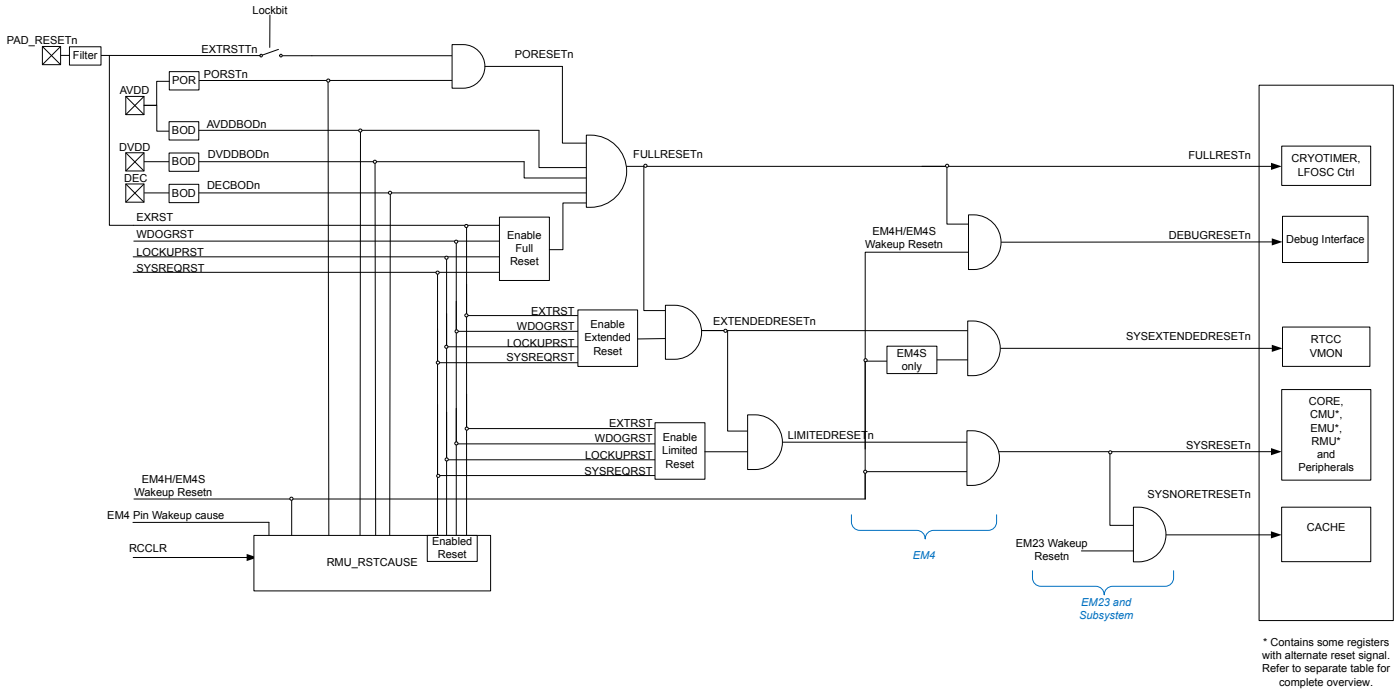


Figure 8.1. RMU Reset Input Sources and Connections

8.3.1 Reset Levels

The reset sources on EFM32 Giant Gecko 12 can be divided in two main groups; Hard resets and Soft resets.

The soft resets can be configured to be either DISABLED, LIMITED, EXTENDED or FULL. The reset level for soft reset sources is configured in the xxxRMODE bitfields in RMU_CTRL.

Table 8.1. Reset Levels

RMU_CTRL_xxxRMODE	Parts of System Reset
DISABLED	Nothing is reset, request will not be registered in RMU_RSTCAUSE
LIMITED	Everything reset, with exception of CRYOTIMER, DEBUGGER, RTCC, VMON and parts of CMU, RMU and EMU.
EXTENDED	Everything reset, with exception of CRYOTIMER, DEBUGGER, and parts of CMU, RMU and EMU.
FULL	Everything reset, with exception of some registers in RMU and EMU.

The reset sources resulting in a soft reset are:

- Watchdog reset
- Lockup reset
- System reset request
- Pin reset (Pin reset can be configured to be either a soft or a hard reset, see 8.3.5 RESETn Pin Reset for details.)

Note: LIMITED and EXTENDED resets are synchronized to HFSRCCLK. If HFSRCCLK is slow, there will be latency on reset assertion. If HFSRCCLK is not running, reset will be asserted after a timeout.

Hard resets will reset the entire chip, the reset sources resulting in a hard reset are:

- Power-on reset
- Brown-out reset
- Pin reset (Pin reset can be configured to be either a soft or a hard reset, see 8.3.5 RESETn Pin Reset for details.)

8.3.2 RMU_RSTCAUSE Register

Whenever a reset source is active, the corresponding bit in the RMU_RSTCAUSE register is set. At startup the program code may investigate this register in order to determine the cause of the reset. The register is cleared upon POR and software write to RMU_CMD_RCCLR. The register should be cleared after the value has been read at startup, otherwise the register may indicate multiple causes for the reset at next startup.

RMU_RSTCAUSE should be interpreted according to Table 8.2 RMU Reset Cause Register Interpretation on page 257. In Table 8.2 RMU Reset Cause Register Interpretation on page 257, the reset causes are ordered by severity from right to left. A reset cause bit is invalidated (i.e. can not be trusted) if one of the bits to the right of it does not match the table. X bits are don't care.

Note: It is possible to have multiple reset causes. For example, an external reset and a watchdog reset may happen simultaneously.

Table 8.2. RMU Reset Cause Register Interpretation

RMU_RSTCAUSE										Reset cause
EM4RST	BUMODERST	WDOGRST	SYSREQRST	LOCKUPRST	EXTRST	DECBOD	DVDDBOD	AVDDBOD	PORST	
X	X	X	X	X	X	X	X	X	1	Power on reset
X	X	X	X	X	X	X	X	1	0	Brown-out on AVDD power
X	X	X	X	X	X	X	1	X	0	Brown-out on DVDD power
X	X	X	X	X	X	1	X	X	0	Brown-out on DEC power
X	X	X	X	X	1	X	X	X	0	Pin reset
X	X	X	X	1	0/X[1]	0	0	0	0	Lockup reset
X	X	X	1	X	0/X[1]	0	0	0	0	System reset request
X	X	1	X	X	0/X[1]	0	0	0	0	Watchdog reset
1	1	X	X	X	0/X[1]	0	0	X[2]	0	System has been in backup mode
1	X	X	X	X	0/X[1]	0	0	0	0	System has been in EM4

1. Pin reset configured as hard/soft
2. AVDDBOD is set to 1 if the reset is caused by a BOD on BUVDD. AVDDBOD will not be set on AVDD recovery or a hard pin reset.

8.3.3 Power-On Reset (POR)

The POR ensures that the EFM32 Giant Gecko 12 does not start up before the AVDD supply voltage has reached the threshold voltage VPORthr (roughly 1.2V). Before the POR threshold voltage is reached, the EFM32 Giant Gecko 12 is kept in reset state. The operation of the POR is illustrated in Figure 8.2 RMU Power-on Reset Operation on page 258, with the active low POWERONn reset signal. The reason for the “unknown” region is that the corresponding supply voltage is too low for any reliable operation.

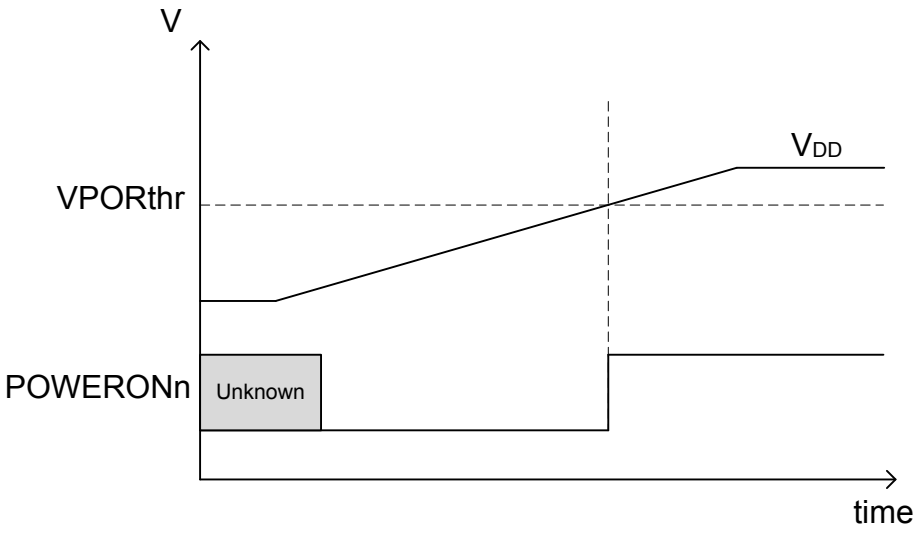


Figure 8.2. RMU Power-on Reset Operation

8.3.4 Brown-Out Detector (BOD)

The EFM32 Giant Gecko 12 has 3 brownout detectors, one for the unregulated power (DVDD), one for the regulated internal power (DECOUPLE), and one for the Analog Power Domain (AVDD). The BODs are constantly monitoring these supply voltages. Whenever the unregulated or regulated power drops below the VBODthr value (see the Electrical Characteristics section of the data sheet for details), or if AVDD drops below the voltage at the DECOUPLE pin, the corresponding active low BROWNOUTn line is held low. The BODs also include hysteresis, which prevents instability in the corresponding BROWNOUTn line when the supply is crossing the VBODthr limit or the AVDD supply drops below the DECOUPLE pin. The operation of the BOD is illustrated in Figure 8.3 RMU Brown-out Detector Operation on page 258. The “unknown” regions are handled by the POR module.

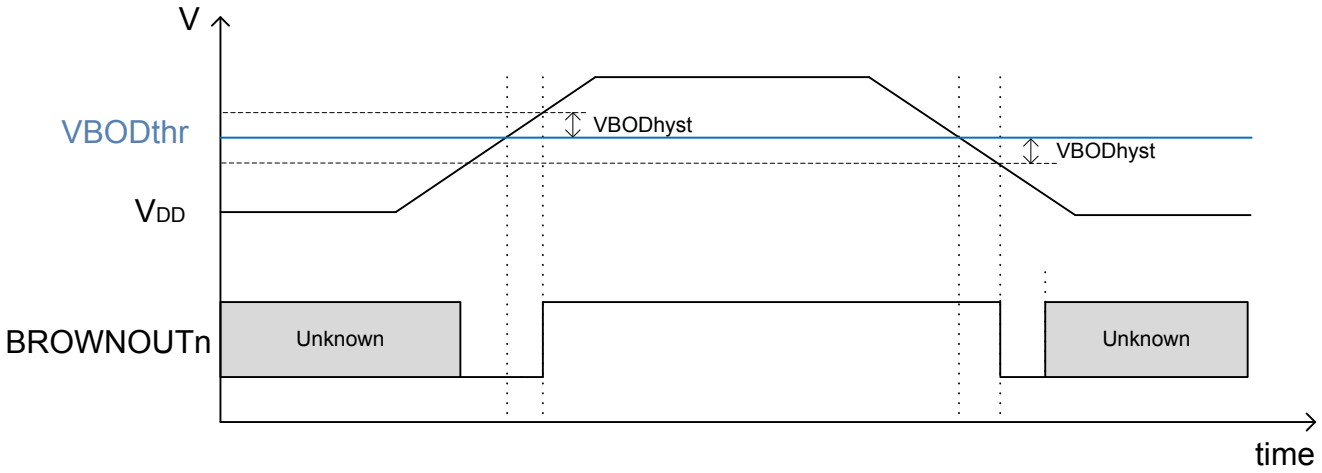


Figure 8.3. RMU Brown-out Detector Operation

8.3.5 RESETn Pin Reset

The pin reset on EFM32 Giant Gecko 12 can be configured to be either hard or soft. By default, pin reset is configured as a soft reset source. To configure it as a hard reset, clear the PINRESETSOFT bit in CLW0 in the Lock bit page, see 6.3.2 Lock Bits (LB) Page Description for details. Forcing the RESETn pin low generates a reset of the EFM32 Giant Gecko 12. The RESETn pin includes an on-chip pull-up resistor, 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 EFM32 Giant Gecko 12.

8.3.6 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. The Watchdog reset can be configured to cause different levels of reset as determined by WDOGRMODE in the RMU_CTRL register.

8.3.7 Lockup Reset

A Cortex-M4 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.

A Cortex-M4 lockup gives immediate indication of seriously errant kernel software. This is the result of the core being locked up due to an unrecoverable exception following the activation of the processor's built in system state protection hardware. For more information about the Cortex-M4 lockup conditions see the ARMv7-M Architecture Reference Manual. The Lockup reset does not reset the Debug Interface, unless configured as a FULL reset. The Lockup reset can be configured to cause different levels of reset as determined by the LOCKUPRMODE bits in the RMU_CTRL register. This includes disabling the reset.

8.3.8 System Reset Request

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. The SYSRESETREQ does not reset the Debug Interface, unless configured as a FULL reset. The SYSRESTREQ reset can be configured to cause different levels of reset as determined by SYSRESETRMODE bits in the RMU_CTRL register. This includes disabling the reset.

8.3.9 Reset State

The RESETSTATE bitfield in RMU_CTRL is a read-write register intended for software use only, and can be used to keep track of state throughout a reset. This bitfield is only reset by POR and hard pin reset.

8.3.10 Register Reset Signals

Figure 8.1 RMU Reset Input Sources and Connections on page 255 shows an overview of how the different parts of the design are affected by the different levels of reset. For RMU, EMU and CMU there are some exceptions. These are given in the following tables.

8.3.10.1 Registers With Alternate Reset

Table 8.3. Alternate Reset for Registers in RMU

RMU Reset Levels	
POR and hard pin reset	RMU_CTRL_WDOGRMODE
	RMU_CTRL_LOCKUPRMODE
	RMU_CTRL_SYSRMODE
	RMU_CTRL_PINRMODE
	RMU_CTRL_RESETSTATE
FULL reset	RMU_LOCK_LOCKKEY

Table 8.4. Alternate Reset for Registers in CMU

CMU Reset Levels	
FULL reset	CMU_LFRCOCTRL
	CMU_LFXOCTRL
EXTENDED reset	CMU_LFECLKSEL
	CMU_LFECLKEN0
	CMU_LFEPRESC0

Table 8.5. Alternate Reset for Registers in EMU

EMU Reset Levels	
POR, BOD, and hard pin reset	EMU_BIASCONF_LSBIAS_SEL
POR, BOD, and hard pin reset	EMU_DCDCLNVCTRL
POR and hard pin reset	EMU_CTRL_EM2BODDIS
	EMU_BUCTRL
	EMU_R5VCTRL
	EMU_R5VADCCTRL
	EMU_R5VDETCTRL_VREGIDETDIS
	EMU_R5VDETCTRL_VBUSDETDIS
	EMU_R5VDETCTRL_VREGODETDIS

EMU Reset Levels	
POR, BOD, and hard pin reset	EMU_PWRCTRL EMU_DCDCCTRL EMU_DCDCMISCCTRL EMU_DCDCZDETCTRL EMU_DCDCCLIMCTRL EMU_DCDCLNCOMPCTRL EMU_DCDCLPVCTRL EMU_DCDCLPCTRL EMU_DCDCLNFREQCTRL EMU_DCDCLPEM01CFG
EXTENDED reset	EMU_VMONAVDDCTRL EMU_VMONALTAVDDCTRL EMU_VMONDVDDCTRL EMU_VMONIO0CTRL EMU_VMONIO1CTRL EMU_VMONBUVDDCTRL
FULL reset	EMU_EM4CTRL EMU_R5VOUTLEVEL

8.4 Register Map

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

Offset	Name	Type	Description
0x000	RMU_CTRL	RW	Control Register
0x004	RMU_RSTCAUSE	R	Reset Cause Register
0x008	RMU_CMD	W1	Command Register
0x00C	RMU_RST	RW	Reset Control Register
0x010	RMU_LOCK	RWH	Configuration Lock Register

8.5 Register Description

8.5.1 RMU_CTRL - 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				0x2				0x0				0x4		
Access							RW											RW				RW				RW				RW		
Name							RESETSTATE											PINRMODE				SYSRMODE				LOCKUPRMODE				WDOGRMODE		

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	RESETSTATE	0x0	RW	**System Software Reset State**
	Bit-field for software use only. This field has no effect on the RMU and is reset by power-on reset and hard pin reset only.			
23:15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:12	PINRMODE	0x4	RW	**PIN Reset Mode**
	Controls the reset level for Pin reset request. These settings only apply when PINRESETSOFT in CLW0 in the Lock bit page is set.			

Value	Mode	Description
0	DISABLED	Reset request is blocked.
1	LIMITED	The CRYOTIMER, DEBUGGER, RTCC, are not reset.
2	EXTENDED	The CRYOTIMER, DEBUGGER are not reset. RTCC is reset.
4	FULL	The entire device is reset except some EMU and RMU registers.

11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	SYSRMODE	0x2	RW	**Core Sysreset Reset Mode**
	Controls the reset level for Core SYSREST reset request.			

Value	Mode	Description
0	DISABLED	Reset request is blocked.
1	LIMITED	The CRYOTIMER, DEBUGGER, RTCC, are not reset.
2	EXTENDED	The CRYOTIMER, DEBUGGER are not reset. RTCC is reset.
4	FULL	The entire device is reset except some EMU and RMU registers.

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

Bit	Name	Reset	Access	Description
6:4	LOCKUPRMODE	0x0	RW	**Core LOCKUP Reset Mode**

Controls the reset level for Core LOCKUP reset request.

Value	Mode	Description
0	DISABLED	Reset request is blocked.
1	LIMITED	The CRYOTIMER, DEBUGGER, RTCC, are not reset.
2	EXTENDED	The CRYOTIMER, DEBUGGER are not reset. RTCC is reset.
4	FULL	The entire device is reset except some EMU and RMU registers.

3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	WDOGRMODE	0x4	RW	**WDOG Reset Mode**

Controls the reset level for WDOG reset request.

Value	Mode	Description
0	DISABLED	Reset request is blocked. This disable bit is redundant with enable/disable bit in WDOG
1	LIMITED	The CRYOTIMER, DEBUGGER, RTCC, are not reset.
2	EXTENDED	The CRYOTIMER, DEBUGGER are not reset. RTCC is reset.
4	FULL	The entire device is reset except some EMU and RMU registers.

8.5.2 RMU_RSTCAUSE - Reset Cause 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																0				0	0	0	0	0				0	0	0		0
Access																R				R	R	R	R	R				R	R	R		R
Name																EM4RST				BUMODERST	WDOGRST	SYSREQRST	LOCKUPRST	EXTRST				DECBOD	DVDDBOD	AVDDBOD		PORST

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	EM4RST	0	R	**EM4 Reset**
	Set if the system has been in EM4. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	BUMODERST	0	R	**Backup Mode Reset**
	Set if the system has been in Backup mode. Must be cleared by software. See EMU chapter for details on how to interpret this bit.			
11	WDOGRST	0	R	**Watchdog Reset**
	Set if a watchdog reset has been performed. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.			
10	SYSREQRST	0	R	**System Request Reset**
	Set if a system request reset has been performed. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.			
9	LOCKUPRST	0	R	**LOCKUP Reset**
	Set if a LOCKUP reset has been requested. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.			
8	EXTRST	0	R	**External Pin Reset**
	Set if an external pin reset has been performed. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	DECBOD	0	R	**Brown Out Detector Decouple Domain Reset**
	Set if a regulated domain brown out detector reset has been performed. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.			
3	DVDDBOD	0	R	**Brown Out Detector DVDD Reset**
	Set if a unregulated domain brown out detector reset has been performed. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.			
2	AVDDBOD	0	R	**Brown Out Detector AVDD Reset**
	Set if a unregulated domain brown out detector reset has been performed. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.			

Bit	Name	Reset	Access	Description
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	PORST	0	R	**Power on Reset** Set if a power on reset has been performed. Must be cleared by software. See 8.3.2 RMU_RSTCAUSE Register for details on how to interpret this bit.

8.5.3 RMU_CMD - Command 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																																0
Access																																W1
Name																																RCCLR

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	RCCLR	0	W1	**Reset Cause Clear** Set this bit to clear the RSTCAUSE register.

8.5.4 RMU_RST - Reset 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																																
Access																																
Name																																

Bit	Name	Reset	Access	Description
31:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

8.5.5 RMU_LOCK - Configuration Lock Register

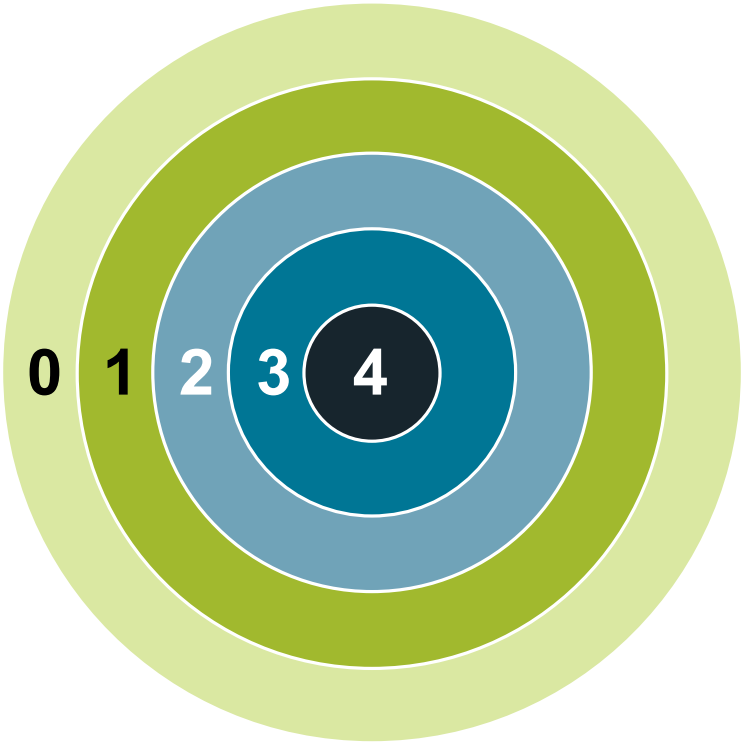
Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0000															
Access																	RWH															
Name																	LOCKKEY															

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	0x0000	RWH	**Configuration Lock Key**

Write any other value than the unlock code to lock RMU_CTRL and RMU_RST from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.

Mode	Value	Description
Read Operation		
UNLOCKED	0	RMU registers are unlocked
LOCKED	1	RMU registers are locked
Write Operation		
LOCK	0	Lock RMU registers
UNLOCK	0xE084	Unlock RMU registers

9. EMU - Energy Management Unit



Quick Facts

What?

The EMU (Energy Management Unit) handles the different low energy modes in EFM32 Giant Gecko 12

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 DeepSleep, and short wake-up time (2 µs from EM2 DeepSleep and EM3 Stop), applications can dynamically minimize energy consumption during program execution.

9.1 Introduction

The Energy Management Unit (EMU) manages all the low energy modes (EM) in EFM32 Giant Gecko 12. Each energy mode manages whether the CPU and the various peripherals are available. The energy modes range from EM0 Active to EM4 Shutoff. EM0 Active mode provides the highest amount of features, enabling the CPU, and peripherals with the highest clock frequency. EM4 Shutoff Mode provides the lowest power state, allowing the part to return to EM0 Active on a wake-up condition. The EMU also controls the various power routing configurations, internal regulators settings, and voltage monitoring needed for optimal power configuration and protection.

9.2 Features

The primary features of the EMU are listed below:

- Energy Modes control
  - Entry into EM4 Hibernate or EM4 Shutoff
  - Configuration of regulators and clocks for each Energy Mode
  - Configuration of various EM4 Hibernate/Shutoff wake-up conditions
  - Configuration of RAM power and retention settings
  - Configuration of GPIO retention settings
- Power routing configurations
  - DCDC control
  - Internal power switches allowing for extensible system power architecture
- Temperature measurement control and status
- Brown Out Detection
- Voltage Monitoring
  - Four dedicated continuous monitor channels
  - Optional monitor features include interrupt generation and low power mode wake-up
- State Retention
- Voltage Scaling
  - EM0/EM1 voltage scaling
  - EM2/EM3 voltage scaling
  - EM4H voltage scaling

9.3 Functional Description

The EMU is responsible for managing the wide range of energy modes available in EFM32 Giant Gecko 12. 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 9.1 EMU Overview on page 269, shows the relative connectivity to the various blocks in the system.

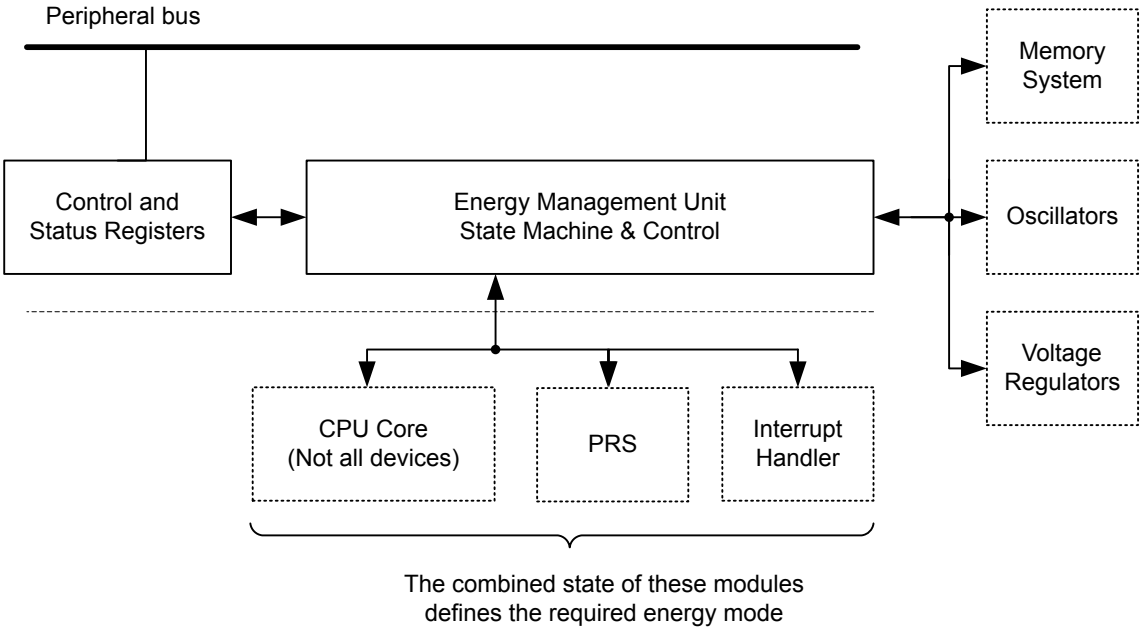


Figure 9.1. EMU Overview

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.

The power architecture is highly configurable to meet system power performance needs. Several external power configurations are supported. The EMU allows flexible control of internal DCDC, Digital LDO Regulator, and internal power switching.

9.3.1 Energy Modes

EFM32 Giant Gecko 12 features six main energy modes, referred to as Energy Mode 0 (EM0 Active) through Energy Mode 4 (EM4 Shutoff). The Cortex-M4 is only available for program execution in EM0 Active. In EM0 Active/EM1 Sleep any peripheral function can be enabled. EM2 DeepSleep through EM4 Shutoff, 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 9.1 table on page 270 shows the possible transitions between different energy modes.

Table 9.1. Energy Mode Transitions

Current Mode	EM Transition Action					
	Enter EM0 Active	Enter EM1 Sleep	Enter EM2 DeepSleep	EnterEM3 Stop	EnterEM4 Hibernate	Enter EM4 Shutoff
EM0 Active		Sleep (WFI, WFE)	Deep Sleep (WFI, WFE)	Deep Sleep (WFI, WFE)	EM4 Entry	EM4 Entry
EM1 Sleep	IRQ		Peripheral wake up done[1]	Peripheral wake up done[1]		
EM2 DeepSleep	IRQ	Peripheral wake up req[1]				
EM3 Stop	IRQ	Peripheral wake up req[1]				
EM4 Hibernate	Wake Up					
EM4 Shutoff	Wake Up					

Note:
1. Peripheral wake-up from EM2/3 to EM1 and then automatically back to EM2/3 when done.

The CSEN, LESENSE, ADC and LEUART have the ability to temporarily wake up the part from either EM2 DeepSleep or EM3 Stop to EM1 Sleep in order to transfer data. Once completed, the part is automatically placed back into the EM2 DeepSleep or EM3 Stop mode.

The Core can always request to go to EM1 Sleep with the WFI or WFE command during EM0 Active. The core will be prevented from entering EM2 DeepSleep or EM3 Stop if Flash is programming or erasing.

An overview of supported energy modes and available functionality is shown in Table 9.2 EMU Energy Mode Overview on page 270. 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 9.2. EMU Energy Mode Overview

	EM0 Active/EM1 Sleep	EM2 DeepSleep	EM3 Stop	EM4 Hibernate	EM4 Shutoff
Wake-up time to EM0 Active/EM1 Sleep	—	2 µs [1]	2 µs [1]	160 µs [1]	160 µs [1]
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]	—	—

	EM0 Active/EM1 Sleep	EM2 Deep-Sleep	EM3 Stop	EM4 Hiber-nate	EM4 Shutoff
High Frequency Oscillators (HFRCO, HFXO) and Clocks (HFSRCLK, HFCLK, HFCORECLK, HFBUSCLK, HFPERCLK, HFPERBCLK, HFPERCCLK, HFCLKLE)	Available	—	—	—	—
Auxiliary High Frequency Oscillator (AUXHFR-CO) and Clock (AUXCLK)	Available	Available[4]	Available[4]	—	—
Low Frequency Oscillators (LFRCO, LFXO)	Available	Available	—	Available	Available
Low Energy Clocks A and B (LFACLK, LFBCLK)	Available	Available	Available[7]	—	—
Low Energy Clock E (LFECLK)	Available	Available	Available[7]	Available	—
ULFRCO (Ultra Low Frequency Oscillator)	On	On	On	On	Available
CRYPTO (Crypto Accelerator)	Available	—	—	—	—
TRNG (True Random Number Generator)	Available	—	—	—	—
GPCRC (Cyclic Redundancy Check)	Available	—	—	—	—
RTCC (Real Time Counter and Calendar)	Available	Available	Available[7]	Available	—
RTCC Memory Retained	Yes	Yes	Yes	Yes	—
USART (USART/SPI)	Available	—	—	—	—
LCD (Liquid Crystal Display)	Available	Available	—	—	—
EBI (External Bus Interface)	Available	—	—	—	—
QSPI (Octal-SPI Flash Controller)	Available	—	—	—	—
USB (Universal Serial Bus)	Available	Available[5]	—	—	—
SDIO	Available	—	—	—	—
CAN (Controller Area Network)	Available	—	—	—	—
LEUART (Low Energy UART)	Available	Available[3]	—	—	—
I^2C	Available	Available[6]	Available[6]	—	—
PDM (Pulse Density Modulation)	Available	—	—	—	—
TIMER (Timer/Counter)	Available	—	—	—	—
LETIMER (Low Energy Timer)	Available	Available	Available[7]	—	—
CRYOTIMER (Ultra Low Energy Timer/Counter)	Available	Available	Available[7]	Available	Available
WDOG (Watchdog)	Available	Available	Available[7]	—	—
PCNT (Pulse Counter)	Available	Available	Available	—	—
CSEN (Capacitive Sense)	Available	Available[3]	—	—	—
ACMP (Analog Comparator)	Available	Available[8]	Available[8]	—	—
ADC (Analog to Digital Converter)	Available	Available[3, 4]	Available[3, 4]	—	—
IDAC (Current Digital to Analog Converter)	Available	Available	Available	—	—
VDAC (Voltage Digital to Analog Converter)	Available	Available	Available	—	—
OPAMP (Operational Amplifier)	Available	Available	Available	—	—

	EM0 Active/EM1 Sleep	EM2 Deep-Sleep	EM3 Stop	EM4 Hiber-nate	EM4 Shutoff
LESENSE (Low Energy Sensor)	Available	Available[3]	—	—	—
EMU Temperature Sensor	Available	Available	Available	Available	—
DC-DC Converter	Available	Available	Available	Available	—
VMON Wake-up or Reset	Available	Available	Available	Available	—
Brown-Out Detect/Power-on Reset	Available	Available	Available	Available	Available
Pin Reset	Available	Available	Available	Available	Available
GPIO Pin Interrupts	Available	Available	Available	Available[9]	Available[9]
GPIO Pin State Retention	Yes	Yes	Yes	Available[10]	Available[10]

Note:

1. Approximate time. Refer to the data sheet
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., ADC, LEUART, LESENSE, CSEN) in EM2/3. Features required by the LDMA which are not supported in EM2/3 (e.g., HFCLK), will be automatically enabled prior to the LDMA transfer and then automatically disabled afterwards.
4. While in EM2/3, an asynchronous event can be routed through PRS (e.g. GPIO IRQ or ACMP output) to wake up the ADC. Features required by the ADC which are not supported in EM2/3 (e.g., AUXHFRCO) will be automatically enabled to allow the ADC to convert a sample, and then automatically disabled afterwards.
5. USB can gate AHB clock and follow partial power-down sequence before entering EM2. Some of the USB blocks are still alive in EM2 in order to detect resume, remote wakeup, SRP, or new session start event, then wake the core.
6. I2C functionality limited to receive address recognition
7. Must be using ULFRCO
8. ACMP functionality in EM2/3 limited to edge interrupt
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.

The different energy modes are summarized in the following sections.

9.3.1.1 EM0 Active

EM0 Active provides all system features.

- Cortex-M4 is executing code
- High and low frequency clock trees are active
- All oscillators are available
- All peripheral functionality is available

9.3.1.2 EM1 Sleep

EM1 Sleep disables the core but leaves the remaining system fully available.

- Cortex-M4 is in sleep mode. Clocks to the core are off
- High and low frequency clock trees are active
- All oscillators are available
- All peripheral functionality is available

9.3.1.3 EM2 DeepSleep

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-M4 is in sleep mode. Clocks to the core are off.
- High frequency clock tree is inactive
- Low frequency clock tree is active
- The following oscillators are available
  - LFRCO, LFXO, ULFRCO, AUXHFRCO (on demand, if used by the ADC)
- The following low frequency peripherals are available
  - RTCC, LCD , WDOG, LEUART, LETIMER, LESENSE, PCNT, CRYOTIMER
- The following analog peripherals are available (with potential limitations on functionality)
  - ADC, IDAC, VDAC, OPAMP, CSEN
- Wake-up to EM0 Active through
  - Peripheral interrupt, reset pin, power on reset, asynchronous pin interrupt, I2C address recognition, or ACMP edge interrupt
- 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
- The DC-DC converter can be configured to remain on in Low Power mode.

9.3.1.4 EM3 Stop

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-M4 is in sleep mode. Clocks to the core are off.
- 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, AUXHFRCO (on demand, if used by the ADC)
- The following low frequency peripherals are available if clocked by the ULFRCO
  - RTCC, WDOG, CRYOTIMER
- The following analog peripherals are available (with potential limitations on functionality)
  - ADC, IDAC, VDAC, OPAMP, CSEN
- Wake-up to EM0 Active through
  - Peripheral interrupt, reset pin, power on reset, asynchronous pin interrupt, I2C address recognition, or ACMP edge interrupt
- 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
- The DC-DC converter can be configured to remain on in Low Power mode.

9.3.1.5 EM4 Hibernate

The majority of peripherals are shutoff to reduce leakage power. A few selected peripherals are available. System memory and registers do not retain values. GPIO PAD state and RTCC RAM are retained. Wake-up from EM4 Hibernate requires a reset to the system, returning it back to EM0 Active

- Cortex-M4 is off
- High frequency clock tree is off
- Some low frequency clock trees may be active
- The following oscillators are available
  - LFRCO, LFXO, ULFRCO
- The following low frequency peripherals are available
  - RTCC, CRYOTIMER
- Wake-up to EM0 Active through
  - VMON, EMU Temperature Sensor, RTCC, CRYOTIMER, reset pin, power on reset, asynchronous pin interrupt (on GPIO_EM4WUx pins only)
- GPIO pin state may be retained (depending on EMU->EM4CTRL.EM4IORETMODE configuration)
- RTCC memory is retained
- The DC-DC converter can be configured to remain on in Low Power mode.

9.3.1.6 EM4 Shutoff

EM4 Shutoff is the lowest energy mode of the part. There is no retention except for GPIO PAD state. Wake-up from EM4 Shutoff requires a reset to the system, returning it back to EM0 Active

- Cortex-M4 is off
- High frequency clock tree is off
- Low frequency clock tree may be active
- The following oscillators are available
  - LFRCO, LFXO, ULFRCO
- The following low frequency peripherals are available
  - CRYOTIMER
- Wake-up to EM0 Active through
  - CRYOTIMER, reset pin, power on reset, asynchronous pin interrupt (on GPIO_EM4WUx pins only)
- GPIO pin state may be retained (depending on EMU->EM4CTRL.EM4IORETMODE configuration)
- The DC-DC converter configuration is reset to its default Unconfigured configuration (DC-DC converter disabled and bypass switch is off)

9.3.2 Entering Low Energy Modes

The following sections describe the requirements for entering the various energy modes.

Note: If Voltage scaling is being used to save system energy, it is important to ensure the proper conditions for entry and exit of EM2 DeepSleep, EM3 Stop or EM4 Hibernate be met. See 9.3.9.2.1 EM2/EM3 Voltage Scaling Guidelines and 9.3.9.3.1 EM4H Voltage Scaling Guidelines for details.

9.3.2.1 Entry Into EM1 Sleep

Energy mode EM1 Sleep is entered when the Cortex-M4 executes the Wait For Interrupt (WFI) or Wait For Event (WFE) instruction while the SLEEPDEEP bit the Cortex-M4 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-M4 System Control Register is set. Refer to ARM documentation on entering Sleep modes.

Alternately, EM1 Sleep can be entered from either EM2 DeepSleep or EM3 Stop from a Peripheral Wake-up Request allowing transfers between the Peripheral and System RAM or Flash. On EFM32, ADC, CSEN, IDAC, LESENSE, and LEUART peripherals can request this wake-up event. Refer to their respective register specification to enable this option. The system will return back to EM2 DeepSleep or EM3 Stop once the ADC, CSEN, IDAC, LESENSE, or LEUART have completed its transfers and processing.

9.3.2.2 Entry Into EM2 DeepSleep or EM3 Stop

Energy mode EM2 DeepSleep or EM3 Stop may be entered when **all** of the following conditions are true:

- IDAC is currently not updating output.
- Cortex-M4 (if present) is in DEEPSLEEP state
- Flash Program/Erase Inactive
- DMA done with all current requests
- A debugger is not currently connected.

Entry into EM2 DeepSleep and EM3 Stop can be blocked by setting the EMU_CTRL->EM2BLOCK bit.

Note: When EM2 DeepSleep or EM3 Stop entry is blocked, the part is not able to enter a lower energy state. The core will be in a sleep state, similar to EM1, where it is waiting for a proper interrupt of other valid wake-up event. Once the blocking conditions are removed, then the part will automatically enter a lower energy state.

Energy mode EM2 DeepSleep is entered from EM0 Active when the Cortex-M4 executes the Wait For Interrupt (WFI) or Wait For Event (WFE) instruction while the SLEEPDEEP bit in the Cortex-M4 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-M4 System Control Register is set. Refer to ARM documentation on entering Sleep modes.

9.3.2.3 Entry Into EM4 Hibernate or EM4 Shutoff

Energy mode EM4 Hibernate and EM4 Shutoff is entered through register access.

Software must ensure no modules are active when entering EM4 Hibernate/Shutoff. EM4CTRL->EM4STATE field must be configured to select either Hibernate (EM4H) or Shutoff (EM4S) mode prior to entering EM4.

Software may enter EM4 Hibernate/Shutoff from EM0 Active 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 Hibernate/Shutoff by software request.

An active debugger connection will prevent entry into EM4 Hibernate/Shutoff.

Note that upon entry into EM4 Shutoff, the DC-DC converter configuration is reset to its default (i.e. Unconfigured) configuration. In the Unconfigured configuration, the DC-DC converter will be disabled and the bypass switch will be turned off.

9.3.3 Exiting a Low Energy Mode

A system in EM2 DeepSleep and EM3 Stop can be woken up to EM0 Active through regular interrupt requests from active peripherals. Since state and RAM retention is available, the EFM32 is fully restored and can continue to operate as before it went into the Low Energy Mode.

Wake-Up from EM4 Hibernate or EM4 Shutoff is performed through reset. Wake-Up from a specific module must be enabled in that module's EM4WUEN register.

Enabled interrupts that can cause wake-up from a low energy mode are shown in Table 9.3 EMU Wake-Up Triggers from Low Energy Modes on page 276. The wake-up triggers always return the EFM32 to EM0 Active/EM1 Sleep. Additionally, any reset source will return to EM0 Active. VMON-based EM4 Hibernate wake-ups also set the corresponding rise or fall interrupt flag. These flags serve as the wake-up source for EM4 Hibernate and must be cleared by software on EM4 Hibernate exit. Not doing so will result in an immediate wake-up after next EM4 Hibernate entry.

Table 9.3. EMU Wake-Up Triggers from Low Energy Modes

Peripheral	Wake-Up Trigger	EM2 Deep-Sleep	EM3 Stop	EM4 Hiber-nate	EM4 Shut-off
LEUART (Low Energy UART)	Receive / transmit	Yes	—	—	—
LETIMER	Any enabled interrupt	Yes	—	—	—
WDOG	Any enabled interrupt	Yes	Yes	—	—
LESENSE	Any enabled interrupt	Yes	—	—	—
LFXO	Ready Interrupt	Yes	—	—	—
LFRCO	Ready Interrupt	Yes	—	—	—
LCD	Any enabled interrupt	Yes	—	—	—
USB	VBUS and charge detect interrupts	Yes	—	—	—
I^2C	Receive address recognition	Yes	Yes	—	—
ACMP	Any enabled edge interrupt	Yes	Yes	—	—
ADC	SINGLE / SCAN FIFO events, window comparator, and VREF overvoltage	Yes	Yes	—	—
CSEN	Wake on threshold	Yes	Yes	—	—
VDAC	Any enabled interrupt except EM23ERRIF	Yes	Yes	—	—
PCNT	Any enabled interrupt	Yes	Yes[1]	—	—
RTCC	Any enabled interrupt	Yes	Yes	Yes[2]	
VMON	Rising or falling edge on any monitored power	Yes	Yes	Yes[2]	—
EMU Temperature Sensor	Measured temperature outside the defined limits	Yes	Yes	Yes[2]	—
CRYOTIMER	Timeout	Yes	Yes	Yes[2]	Yes[2]
Pin Interrupts	Transition	Yes	Yes	Yes[2, 3]	Yes[2, 3]
Reset Pin	Assertion	Yes	Yes	Yes	Yes
Power	Cycle Off/On	Yes	Yes	Yes	Yes

Peripheral	Wake-Up Trigger	EM2 Deep-Sleep	EM3 Stop	EM4 Hiber-nate	EM4 Shut-off

Note:

1. When using an external clock
2. Corresponding bit in the module's EM4WUEN must be set.
3. Only available on a subset of the pins. Refer to the data sheet for details.

9.3.4 Power Configurations

The EFM32 Giant Gecko 12 allows several power configurations with additional options giving flexible power architecture selection.

In order to provide the lowest power consuming solutions, the EFM32 Giant Gecko 12 comes with a DC-DC module to power internal circuits. The DC-DC requires an external inductor and capacitor (refer to the data sheet for recommended values).

The EFM32 Giant Gecko 12 has multiple internal power domains: IO Supply (IOVDD0,IOVDD1,IOVDD2), Analog & Flash (AVDD), Input to Digital LDO (DVDD), and Low Voltage Digital Supply (DECOUPLE). Additional detail for each configuration and option is given in the following sections.

When assigning supply sources, the following requirement must be adhered to:

- VREGVDD = AVDD (Must be the highest voltage in the system)
- VREGVDD >= DVDD
- VREGVDD >= IOVDD0,IOVDD1,IOVDD2

9.3.4.1 Power Configuration 0: Unconfigured

Upon power-on reset (POR) or entry into EM4 Shutoff, the system is configured in a safe state that supports all of the available Power Configurations. The Unconfigured Configuration is shown in the simplified diagram below.

In the Unconfigured Configuration:

- The DC-DC converter's Bypass switch is OFF.
- The internal digital LDO is powered from the AVDD pin (i.e. REGPWRSEL=0 in EMU_PWRCTRL). Note the maximum allowable current into the LDO when REGPWRSEL=0 is 20 mA. For this reason, immediately after startup firmware should configure REGPWRSEL=1 to power the digital LDO from DVDD.
- The analog blocks are powered from the AVDD supply pin (i.e., ANASW=0 in EMU_PWRCTRL).

After power on, firmware can configure the device to based on the external hardware configuration. Note that the PWRCFG register can only be written once to a valid value and is then locked. This should be done immediately out of boot to select the proper power configuration. The DC-DC and PWRCTRL registers will be locked until the PWRCFG register is configured.

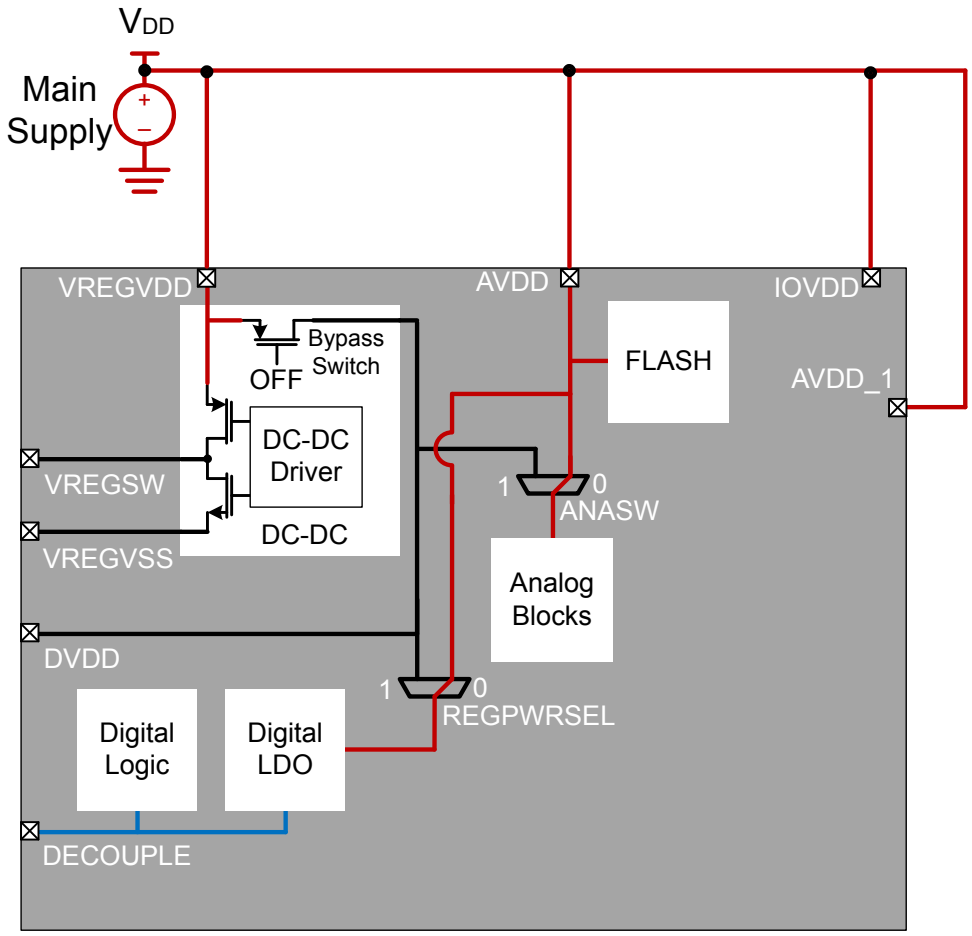


Figure 9.2. Unconfigured Power Configuration

9.3.4.2 Power Configuration 1: No DC-DC

In Power Configuration 1, the DC-DC converter is programmed in Off mode and the Bypass switch is Off. The DVDD pin must be powered externally - typically, DVDD is connected to the main supply. DVDD powers the internal Digital LDO (i.e., REGPWRSEL=1) which powers the digital circuits. IOVDD and AVDD are powered from the main supply as well.

VREGSW must be left disconnected in this configuration.

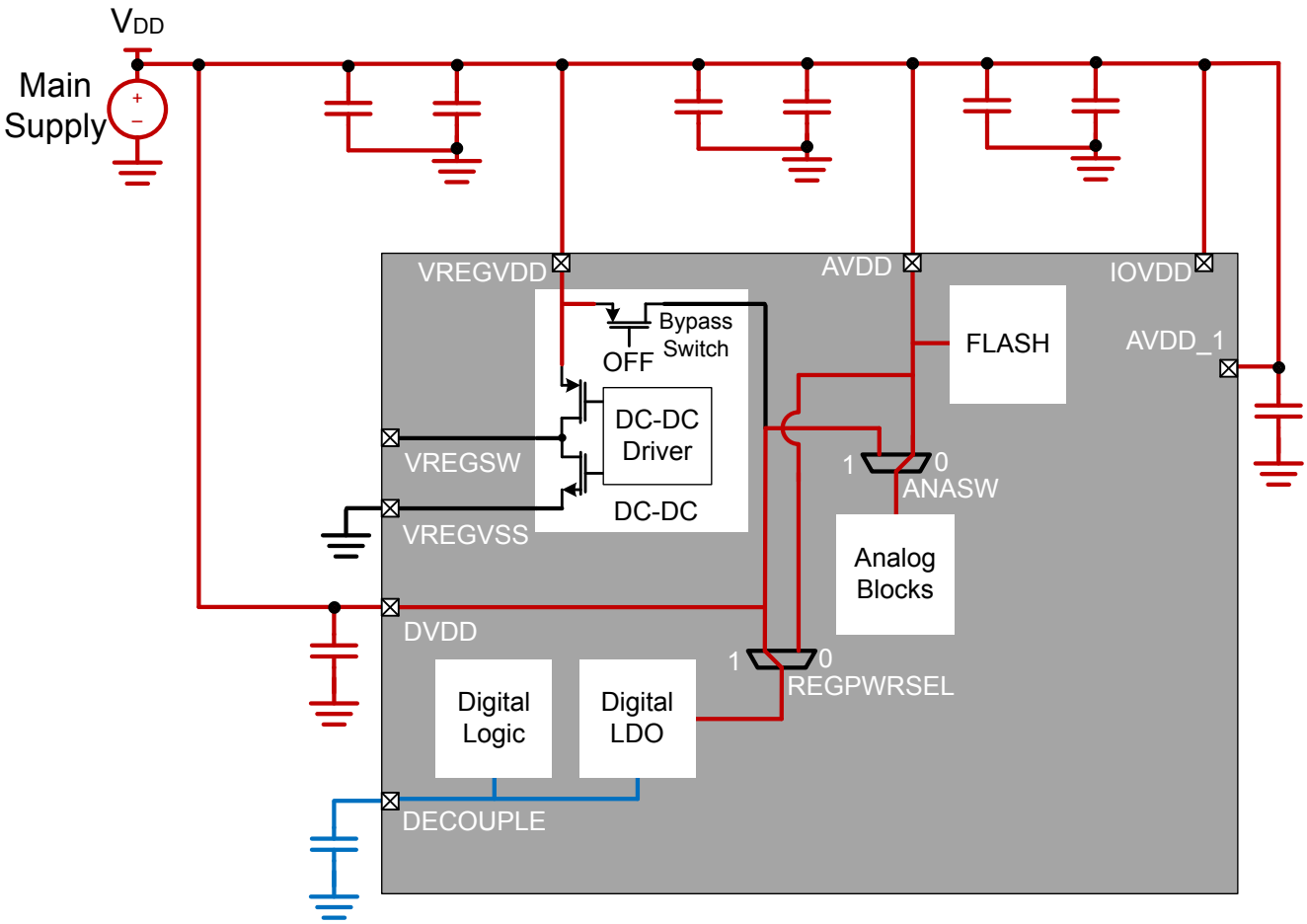


Figure 9.3. DC-DC Off Power Configuration

9.3.4.3 Power Configuration 2: DC-DC

For the lowest power applications, the DC-DC converter can be used to power the DVDD supply.

In Power Configuration 2, the DC-DC Output (V_{DCDC}) is connected to DVDD. DVDD powers the internal Digital LDO (i.e., REGPWRSEL=1) which powers the digital circuits. AVDD is connected to the main supply voltage. The internal analog blocks may be powered from AVDD or DVDD, depending on the ANASW configuration.

IOVDD could be connected to either the main supply (as shown below) or to V_{DCDC}, depending on the system IO requirements. Because V_{DCDC} will be unpowered (i.e., floating) at startup, if IOVDD is powered from the DC-DC converter then any circuit attached to IOVDD will not be powered until the DC-DC is configured (or the bypass switch is enabled). Refer to 9.3.8 IOVDD Connection section for further details and issues that may result when connecting IOVDD to V_{DCDC}.

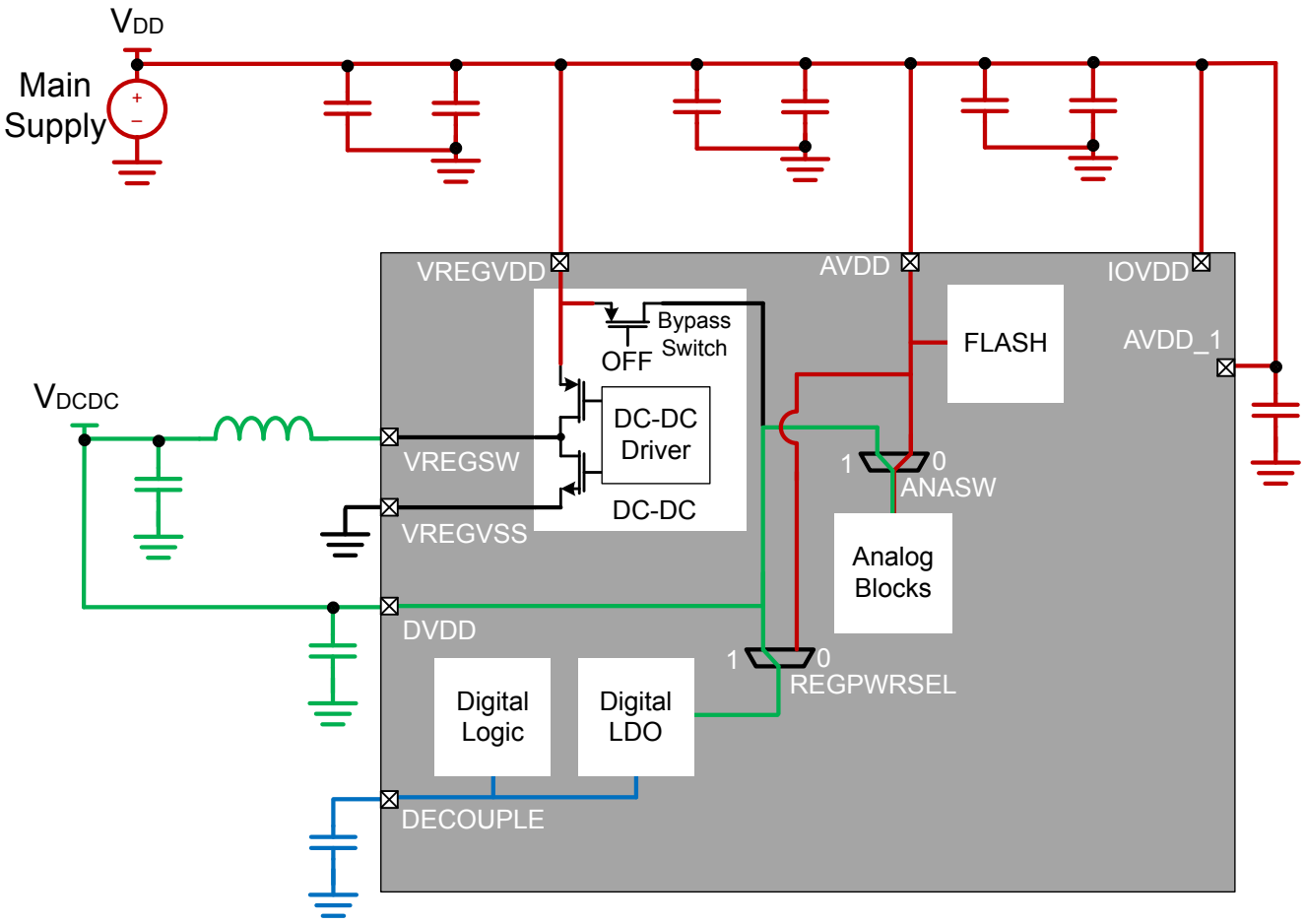


Figure 9.4. DC-DC Standard Power Configuration

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.

9.3.5 DC-to-DC Interface

The EFM32 Giant Gecko 12 devices feature a DC-DC buck converter which requires a single external inductor and a single external capacitor. The converter takes the VREGVDD input voltage and converts it down to an output voltage between VREGVDD and 1.8 V with a peak efficiency of approximately 90% in Low Noise (LN) mode and 85% in Low Power (LP) mode. Refer to the data sheet for full DC-DC specifications.

The DC-DC converter operates in either Low Noise (LN) or Low Power (LP) mode. LN mode is intended for higher current operation (e.g., >= 10 mA), whereas LP mode is intended for very low current operation (e.g., < 10 mA).

In addition, the DC-DC converter supports an unregulated Bypass mode, in which the input voltage is directly shorted to the DC-DC output.

9.3.5.1 Bypass Mode

In Bypass mode, the VREGVDD input voltage is directly shorted to the DC-DC converter output through an internal switch. Consult the data sheet for the Bypass switch impedance specification.

The Bypass Current Limit limits the maximum current drawn from the input supply in Bypass mode. This current limit is enabled by setting the BYPLIMEN bit in the EMU_DCDCCLIMCTRL register, and the limit value may be adjusted between 20 mA and 320 mA using the BYPLIMSEL bitfield in the EMU_DCDCMISCCTRL register. When the difference between the DC-DC output voltage (V_{DCDC}) and the DC-DC input voltage (VREGVDD) is large, applications should enable the Bypass Current Limit before enabling Bypass mode. For example, if Bypass mode is enabled with VREGVDD=3.8 V and V_{DCDC}=1.8 V with a 4.7 µF capacitor, the peak current draw may be quite large as it is limited only by the bypass switch on-resistance, which could result in drooping on the input supply voltage. For smaller input / output voltage differences (e.g., VREGVDD=2.4 V and V_{DCDC}=1.8 V), it may not be necessary to enable the Bypass Current Limit at all.

Note that the device will see an additional ~10 µA of current draw when both the Bypass Current Limiter and Bypass Mode are enabled. Applications should therefore disable the Bypass Current Limiter (i.e., set BYPLIMEN = 0) after the DVDD voltage has reached the main supply voltage in Bypass Mode.

9.3.5.2 Low Power (LP) Mode

The Low Power (LP) controller operates in a hysteretic mode to keep the output voltage within a defined voltage band. Once the DC-DC output voltage drops below a programmable internal reference, the LP controller generates a pulse train to control the powertrain PFET switch, which charges up the DC-DC output capacitor. When the output voltage is at the programmed upper level, the powertrain PFET is turned off. The output ripple voltage may be quite large (>100 mV) in LP mode.

The LP controller supports load currents up to approximately 10 mA, making it suitable for light loads in EM0 and EM1, as well as EM2, EM3, or EM4 low energy modes.

9.3.5.3 Low Noise (LN) Mode

The Low Noise (LN) controller continuously switches the powertrain NFET and PFET switches to maintain a constant programmed voltage at the DVDD pin. The LN controller supports load current from sub-mA up to 200 mA.

The LN controller switching frequency is programmable using the RCOBAND bitfield in the EMU_DCDCLNFREQCTRL register. See below for recommended RCOBAND settings for each mode.

The DC-DC Low Noise controller operates in one of two modes:

1. Continuous Conduction Mode (CCM)
2. Discontinuous Conduction Mode (DCM)

9.3.5.3.1 Low Noise (LN) Continuous Conduction Mode (CCM)

CCM operation is configured by setting the LNFORCECCM bit in the EMU_DCDCMISCCTRL register. CCM can be used to improve the DC-DC converter's output transient response time to quick load current changes, which minimizes voltage transients on the DC-DC output.

Note that all references to CCM in the documentation actually refer to Forced Continuous Conduction Mode (FCCM) - that is, if the LNFORCECCM bit is set and the output load current is very low, the DC-DC will be forced to operated in CCM. In this case, the current through the inductor may be negative and current may flow back into the battery.

In CCM, the recommended DC-DC converter switching frequency is 6.4 MHz (RCOBAND = 4).

9.3.5.3.2 Low Noise (LN) Discontinuous Conduction Mode (DCM)

To enable DCM, the LNFORCECCM bit in EMU_DCDCMISCCTRL must be cleared before entering LN. Typically, this configuration would occur while the part was in Bypass mode. Once DCM is enabled, the DC-DC should operate in DCM at light load currents. However, as the load current increases, the DC-DC will automatically transition into CCM without software intervention.

The advantage of DCM is improved efficiency for light load currents. However, in DCM the DC-DC has poorer dynamic response to changes in load current, leading to potentially larger changes in the regulated output voltage. For these reasons, DCM is not recommended for applications that expect large instantaneous load current steps. For example, if the DC-DC is in DCM, firmware may need to increment the core clock frequency in small steps to prevent a large sudden load increase.

In DCM, the recommended DC-DC converter switching frequency is 3 MHz (RCOBAND = 0).

9.3.5.4 DC-to-DC Programming Guidelines

Note: Refer to Application Note *AN0948: EFM32 and EFR32 Series 1 Power Configurations and DC-DC*for detailed information on programming the DC-DC. Application Notes can be found on the Silicon Labs website (www.silabs.com/32bit-appnotes) or using the [**Application Notes**] tile in Simplicity Studio.

9.3.6 Analog Peripheral Power Selection

The analog peripherals (e.g., ULFRCO, LFRCO, LFXO, HFRCO, AUXHFRCO, VMON, IDAC, ADC, LCD, CSEN) are powered from an internal analog supply domain, VDDX_ANA. VDDX_ANA may be supplied from either the AVDD or DVDD supply pins, depending on the configuration of the ANASW bit in the EMU_PWRCTRL register. Changes to the ANASW setting should be made immediately out of reset (i.e., in the Unconfigured Configuration), before all clocks (with the exception of HFRCO and ULFRCO) are enabled. If the DCDC converter is used and ANASW is set to 1, the switch will not take effect until after the DCDC output voltage has reached its target level. To prevent supply transients, firmware should configure and enable the DCDC, configure ANASW, and then enable clocks. If the DCDC converter is not used, IMMEDIATEPWRSWITCH should be set prior to setting ANASW so hardware can immediately apply the switch without waiting for the DCDC to settle.

Once ANASW is configured it should not be changed. Note that the flash is always powered from the AVDD pin, regardless of the state of the ANASW bit.

Table 9.4. Analog Peripheral Power Configuration

ANASW	Analog Peripheral Power Supply Source (VDDX_ANA)	Comments
0 (default)	AVDD pin	This configuration may provide a quieter supply to the analog modules, but is less efficient as AVDD is typically at a higher voltage than DVDD.
1	DVDD pin	This configuration may provide a noisier supply to the analog modules, but is more efficient. However, because the maximum allowable input voltage to many of the analog modules using APORT is limited to MIN(VDDX_ANA,IOVDD), this setting could artificially limit your analog input range.

9.3.7 Digital LDO Power Selection

The digital LDO may be powered from one of two supply pins, depending on the configuration of the REGPWRSEL bit in the EMU_PWRCTRL register. At startup, the digital is powered from the AVDD pin. When powered from AVDD, the LDO current is limited to 20 mA. Out of startup, firmware should configure and enable the DCDC (if desired) and then set REGPWRSEL=1 before increasing the core clock frequency.

Table 9.5. Digital LDO Power Configuration

REGPWRSEL	Digital LDO Power Source	Comments
0 (default)	AVDD pin	Maximum LDO current in this configuration is 20 mA. Firmware should configure REGPWRSEL to 1 after startup.
1	DVDD pin	This configuration supports all core frequencies, and should be used after startup.

9.3.8 IOVDD Connection

The IOVDD supply(s) must be less than or equal to AVDD. IOVDD will typically be connected to either the DC-DC Output (V_{DCDC}) or the main supply.

Because V_{DCDC} will be unpowered (i.e., floating) at startup, if IOVDD is powered from the DC-DC converter then any circuit attached to IOVDD will not be powered until the DC-DC is configured (or the bypass switch is enabled).

Note: This constraint can have serious and unintended side-effects. For example, if IOVDD=V_{DCDC}:

1. It isn't directly possible to program an unprogrammed device on a PCB through the serial wire interface. Programming the device requires IOVDD to be present (i.e., for SWCLK, SWDIO, etc), and IOVDD won't be present until after the part is programmed (i.e., the DC-DC is enabled in firmware to power up V_{DCDC}). It is possible to work around this issue, however, by providing an external supply for V_{DCDC} during programming.
2. Some unprogrammed devices are preloaded with a bootloader. The bootloader is expecting to read a logic high on the SWCLK pin to determine if the bootloader should execute. With no valid IOVDD voltage present, the code may incorrectly decide to execute the bootloader, which will cause the system to wait in the bootloader until a reset occurs.

Additionally, upon entry into EM4 Shutoff, the DC-DC converter configuration is reset to its default (Unconfigured) configuration. If IOVDD = V_{DCDC}, then any circuits attached to IOVDD will remain unpowered until the system is reset to exit EM4 Shutoff, and the DC-DC is configured (or the bypass switch is enabled).

Any application with powering external loads from the DC-DC converter must take into consideration the maximum allowable DC-DC load current. Refer to the data sheet for DC-DC load current specification.

9.3.9 Voltage Scaling

The voltage scaling feature allows for a tradeoff between power and performance. Voltage scaling applies an adjustment to the supply voltage for the on-chip digital logic and memories. For EM0 and EM1 operation, full device performance is supported when the Voltage Scale Level is set to its highest value. The Voltage Scale Level may be set lower when operating the system at slower clock speeds to save power. Voltage scaling does not affect the input or output range for analog peripherals or digital I/O logic levels. For more information about max system frequency supported for different voltage scaling levels. Refer to the CMU chapter and the data sheet specification tables.

Note: Some device sub-systems and operations are only supported at Voltage Scale Level 2.

- Flash write/erase is only supported at Voltage Scale Level 2.
- TRNG operation is only supported at Voltage Scale Level 2.

Separate voltage scaling controls are available for the different energy modes. These are as follows:

- EM0/EM1 Voltage Scaling
- EM2/EM3 Voltage Scaling
- EM4H Voltage Scaling

9.3.9.1 EM0/EM1 Voltage Scaling

In energy modes EM0 and EM1, the user can dynamically scale voltages between Voltage Scale Level 2 and Voltage Scale Level 0 using the EM01VSCALE2 and EM01VSCALE0 bitfields in EMU_CMD register. A lower Voltage Scale Level can be used in conjunction with lower processor frequency to reduce power consumption. Once these commands are issued, hardware begins the process of voltage scaling and when done, the VSCALEDONE interrupt is triggered. Users can also poll VSCALEBUSY in EMU_STATUS which indicates that hardware is busy changing the voltage scale setting when set. VSCALE in EMU_STATUS shows the current voltage the system is in at any time.

Note:

- If more than one voltage scaling command is issued in EMU_CMD simultaneously, the lower voltage scaling level has higher priority. e.g. priority order: EM01VSCALE0 > EM01VSCALE2.
- The reset value of VSCALE for EM0 and EM1 operation is Voltage Scale Level 2.

When voltage scaling up or down, the user should follow the following sequences in order to ensure proper scaling.

- Voltage Scale Down
  1. Decrease system clock frequency to the target frequency
  2. Update the wait states of Flash or RAM for the target frequency
  3. Issue voltage scaling command by setting EM01VSCALE2 or EM01VSCALE0 in EMU_CMD
  4. Once Hardware completes voltage scaling up, VSCALEDONE interrupt is set.
- Voltage Scale Up
  1. Issue voltage scaling command by setting EM01VSCALE2 or EM01VSCALE0 in EMU_CMD
  2. Wait for hardware to complete voltage scaling. When done, VSCALEDONE interrupt is set.
  3. Update the wait states of Flash or RAM for the target frequency
  4. Increase system clock frequency to the target frequency

Multiple voltage scaling commands are allowed to be issued even when the current voltage scaling is not yet completed. In such a case, the current scaling will be aborted and the last command will be executed. VSCALEDONE interrupt will be issued for every voltage scaling command.

Note: When a hard reset occurs, VSCALE will be set to the reset value (Voltage Scale Level 2). In most cases, a soft reset will not affect the current VSCALE level. However, when a soft reset is issued in the middle of the voltage scaling process, the minimum voltage scale level indicated by VSCALE or the EMU_CMD which triggered the voltage scale operation will be applied and reflected in VSCALE.

9.3.9.2 EM2/EM3 Voltage Scaling

The EM23VSCALE bitfield in EMU_CTRL allows user to independently setup the voltage scaling value for EM2/EM3 energy mode. The EM23VSCALE in EMU_CTRL should be programmed to a level which is less than or equal to VSCALE in EMU_STATUS. This means that EM2/EM3 voltage scaling is always a voltage scaling down process. If EM23VSCALE level in EMU_CTRL is greater than VSCALE level in EMU_STATUS, the VSCALE level will be implemented in EM2/EM3 instead of EM23VSCALE. Upon EM2/EM3 entry, the system will scale down the voltage to a smaller level between VSCALE or EM23VSCALE.

Note: The reset value of EM23VSCALE is Voltage Scale Level 2. Therefore, if user scales EM0/EM1 voltage to Voltage Scale Level 0 (reflected in VSCALE in EMU_STATUS) and enters EM2/EM3, this VSCALE voltage of Voltage Scale Level 0 is maintained in EM2/EM3 as well since this is smaller level between VSCALE and EM23VSCALE.

9.3.9.2.1 EM2/EM3 Voltage Scaling Guidelines

Note that when using EM23VSCALE in EMU_CTRL to scale down EM2/EM3, the scaled down voltage in EM2/EM3 is maintained after waking from EM2/EM3 to EM0/EM1. For example, if VSCALE was at Voltage Scale Level 2 prior to EM2/EM3 entry, and EM23VSCALE was set to Voltage Scale Level 0, the system will scale down to Voltage Scale Level 0 on EM2/EM3 entry. When waking up to EM0/EM1, the system maintains its voltage at Voltage Scale Level 0. Therefore, user must ensure the system clock frequency and Flash or Ram wait states are programmed to correct values to support waking up to EM0/EM1 at the lower voltages prior to EM2/EM3 entry.

EM23VSCALEAUTOWSEN bitfield in EMU_CTRL enables hardware to automatically configure the system clock frequency and Flash wait states to support low voltage operation when waking up to EM0/EM1 from EM2/EM3. Therefore, this obviates the need for user to setup the clock frequency and Flash wait states prior to EM2/EM3 entry with EM23VSCALE. When waking up to EM0/EM1, while using EM23VSCALEAUTOWSEN set to 1, the HFRCO will default to its production calibrated 19 MHz frequency.

Note: When using EM23VSCALEAUTOWSEN set to 1, the user should still program RAM wait states to support 19 MHz prior to EM2/EM3 entry.

9.3.9.3 EM4H Voltage Scaling

EM4HVSCALE bitfield in EMU_CTRL allows user to independently setup the voltage scaling levels for EM4H energy mode. The EM4HVSCALE in EMU_CTRL should be programmed to a level which is smaller than or equal to VSCALE level in EMU_STATUS or EM23VSCALE in EMU_CTRL. This means that EM4H voltage scaling is always a voltage scaling down process. If EM4HVSCALE level in EMU_CTRL is greater than level of VSCALE in EMU_STATUS or level of EM23VSCALE in EMU_CTRL, the smaller of VSCALE, EM23VSCALE or EM4HVSCALE levels will be implemented in EM4H.

Note: The reset level of EM4HVSCALE is Voltage Scale Level 2. Therefore, if user scales EM0/EM1 voltage to Voltage Scale Level 0 (reflected in VSCALE in EMU_STATUS) and enters EM4H, this VSCALE voltage of Voltage Scale Level 0 is maintained in EM2/EM3 as well since this is minimum of VSCALE and EM23VSCALE.

9.3.9.3.1 EM4H Voltage Scaling Guidelines

Note that when using EM4HVSCALE in EMU_CTRL to scale down voltage in EM4H, the scaled down voltage in EM4H is maintained after waking from EM4H to EM0/EM1. For example prior to EM4H entry, if VSCALE was at Voltage Scale Level 2 and EM4HVSCALE was set to Voltage Scale Level 0, the system will scale down to Voltage Scale Level 0 on EM4H entry. When waking up to EM0/EM1, the system maintains its voltage at Voltage Scale Level 0.

9.3.9.4 Voltage Scaling Recommended Use

Refer to the data sheet for the maximum supported system frequencies for different Voltage Scaling Levels. Use of the lowest voltage scaling level is recommended for maximum power savings. For any voltage scaling level, it is recommend to use the highest frequency for performance benefits.

Voltage can then be scaled to higher voltage scale levels only when higher system clock frequency is required by the application for a period of time after which user can dynamically scale the voltage back to lower voltage scale levels to continue saving power.

9.3.10 EM2/EM3 Peripheral Retention Disable

Peripherals that are available in EM2 DeepSleep or EM3 Stop can optionally be powered down during EM2 DeepSleep or EM3 Stop. This allows lower energy consumption in these energy modes. However, when powering down, these peripherals are independently reset so the registers lose their configuration values. Therefore, they will have to be reconfigured upon wake-up to EM0 Active if they were previously configured to non reset values.

EMU_EM23PERNORETAINCTRL register can be used to setup unused peripherals for powering down prior to EM2/EM3 entry. Once setup, upon EM2/EM3 entry, all peripherals in the power-down domain will get powered down if all of them are setup to be disabled.

Note: User must ensure that the peripherals being powered down should have their clocks disabled in CMU prior to EM2/EM3 entry.

On waking up from EM2/EM3, EMU_EM23PERNORETAINSTATUS register indicates if the peripherals were powered down by the system and subsequently locked out from register access. Locking out peripherals prevents users from accidentally using peripherals with configurations at their reset state. EMU_EM23PERNORETAINCMD allows user to unlock these peripherals and hence grant access to their registers for updating their configurations.

9.3.11 Brown Out Detector (BOD)

The EFM32 Giant Gecko 12 contains multiple supply brown out detectors (BODs).

9.3.11.1 AVDD BOD

The EFM32 Giant Gecko 12 has a fast response BOD on AVDD that is always active. This BOD ensures the minimal supply is provided to the AVDD supply (typically also connected to VREGVDD). Once triggered, the BOD will cause the system to reset.

Note: In EM4 Hibernate/Shutoff a low power version of the AVDD BOD, called EM4BOD, is available to trigger a reset at level lower than in other energy modes. All other BODs are disabled during EM4 Hibernate/Shutoff

9.3.11.2 DVDD and DECOUPLE BOD

Additional BODs will monitor DVDD and DECOUPLE during EM0 Active through EM3 Stop. This can cause a reset to the internal logic, but will not cause a power-on reset or reset the EMU or RTCC.

9.3.12 Voltage Monitor (VMON)

The EFM32 features an extremely low energy Voltage Monitor (VMON) capable of running down to EM4 Hibernate. Trigger points are preloaded but may be reconfigured.

- AVDD X 2
- DVDD
- IOVDD0 and IOVDD1
- BUVDD

Table 9.6. VMON Events

Feature	Condition	AVDD	DVDD	BUVDD	IOVDD
Hysteresis (separate rise and fall triggers)	—	Yes	—	—	—
Supply switch to/from Backup	Fall/Rise	Yes	—	—	—
Interrupt	Fall or Rise	Yes	Yes	Yes	Yes
Wake-Up from EM4 Hibernate	Fall or Rise	Yes	Yes	Yes	Yes

The status of the VMON is reflected in the EMU_STATUS register.

The status of the sticky interrupt can be found at EMU_IF. These interrupt flags also serve as the wake-up source of EM4H when the associated RISEWU and FALLWU bits are set. This means that if these flags are set, EM4H entry will result in an immediate wake-up. To prevent this, these must be cleared by software before EM4H entry.

Note that the VMON has offset high hysteresis, specified in the device Data Sheet. For rising edge detection the threshold will be the threshold setting (as described below) + V_{VMON_HST}, and for falling edge detection the threshold will simply be the threshold setting.

VMON channels are calibrated at two voltages: 1.86 V and 2.98 V. The calibration results (coarse thresholds and fine thresholds for 1.86 V and 2.98 V) are placed in the VMONCAL registers in the DI page. Using these thresholds it is possible to calculate thresholds for the entire supported VMON VDD range, i.e., 1.62 V to 3.4 V. Using the values given in VMONCAL registers, one can calculate $T_{1.86}$, $T_{2.98}$, V_a and V_b.

$$T_{1.86} = (10 \times \text{VMONCALX\_XVDD1V86THRESCOARSE}) + \text{VMONCALX\_XVDD1V86THRESFINE},$$

$$T_{2.98} = (10 \times \text{VMONCALX\_XVDD2V98THRESCOARSE}) + \text{VMONCALX\_XVDD2V98THRESFINE},$$

$$V_a = (1.12) / (T_{2.98} - T_{1.86}),$$

$$V_b = 1.86 - (V_a \times T_{1.86}),$$

Figure 9.5. VMON Calibration Equations

Now if it is required to find the coarse and fine thresholds for a certain voltage Y, following equation can be used:

$$\text{Thres}_Y = (Y - V_b) / V_a,$$

$$Y_{calib} = (\text{Thres}_Y \times V_a) + V_b,$$

Figure 9.6. VMON Threshold Equations

Thres_Y should be rounded to the nearest integer. The least significant digit of the rounded Thres_Y gives the fine threshold and remaining digits give the coarse threshold for Y. These can now be programmed in the relevant EMU_VMONXVDDCTRL register as the coarse and fine thresholds. It may not be possible to set threshold exactly for Y. In that case the closest possible voltage is used. Y_{calib} gives the value of this closest possible voltage.

Consider the example where it is required to set the AVDD rise threshold to 2.2 V (so Y=2.2 V). This means that the EMU_VMONAVDDCTRL_RISETHRESCOARSE and EMU_VMONAVDDCTRL_RISETHRESFINE need to be programmed. Here are the steps that should be followed:

- Check VMONCAL0 register. It has the VMON AVDD channel calibrated thresholds for 1.86 V and 2.98 V. Lets assume that the following values are present in the associated bitfields:
  - AVDD1V86THRESCOARSE = 3
  - AVDD1V86THRESFINE = 5
  - AVDD2V98THRESCOARSE = 8
  - AVDD2V98THRESFINE = 7
- Using the above numbers and the VMON calibration equations:
  - $T_{1.86}$ = 35
  - $T_{2.98}$ = 87
  - V_a = 21.53 mV
  - V_b = 1.106 V
- Using the VMON threshold equations (with Y=2.2 V), $Thres_Y$ = 51 (rounded from 50.8) and Y_{calib} = 2.204 V

EMU_VMONAVDDCTRL_RISETHRESCOARSE should be programmed to 5 and EMU_VMONAVDDCTRL_RISETHRESFINE should be programmed to 1 (since $Thres_Y$ = 51). With these programmed values, VMON AVDD rise threshold is set for Y_{calib} = 2.204 V, which is the closest programmable threshold.

9.3.13 Powering Off SRAM Blocks

SRAM blocks may be powered off using the EMU_RAMxCTRL RAMPOWERDOWN fields. Selected blocks are powered down in order from the highest to lowest address in each bank. The lowest SRAM block in RAM0 cannot be powered off and will always remain powered on for proper system functionality. The stack must be located in retained memory. Refer to the EMU_RAMxCTRL register descriptions for power configuration options and the associated address ranges.

9.3.14 5 V Sub-System

The 5 V sub-system manages the 5 V power domains of a chip which includes a 5 V regulator. The 5 V sub-system has two 5 V input supply pins, VBUS and VREGI. The output is available externally via the VREGO pin. If USB is available in the device, the output of the 5 V sub-system is also responsible for powering up the USB PHY. The 5 V sub-system is available from EM0 Active to all the way down to EM4 Shutoff.

9.3.14.1 External Connection Configurations

The sections below describe the different connection configurations for typical use cases of the 5 V sub-system.

It is recommended to use a 1.0 uF bypass capacitor at the VBUS and VREGI supply input pins to meet the USB inrush current specification. A 4.7 uF capacitor should be connected to the VREGO pin. Using greater than 4.7 uF will result in excess inrush current from the source supply. Internal pulldowns on VBUS, VREGI and VREGO prevent leakage and other anomalies when any of these pins are not being used, and so any of VBUS, VREGI and VREGO can be left disconnected when they are not used. See the device data sheet for details about VBUS, VREGI and VREGO specifications.

9.3.14.1.1 5 V Sub-System Powered Device

One of the typical use cases of the 5 V sub-system is to power a device using the output of the 5 V sub-system. In this use case, either VBUS or VREGI provides power and AVDD is externally shorted to VREGO, so all system power comes from a 5 V supply. On initial power-up, the circuit automatically selects the larger of VBUS and VREGI as the power source for the 5 V regulator. When the 5 V regulator powers up, the VREGO voltage ramps up to 2.4V. After the MCU is fully powered and reset is released, various aspects of the 5 V sub-system are configurable by software through the register interface.

Figure 9.7 5 V Sub-System Powered Device Configuration With VREGI and VBUS Connected to Source Supply on page 289 shows the recommended configuration for powering up a device using the 5 V sub-system with both VREGI and VBUS inputs connected to source supplies. This type of configuration can also be used in applications that are USB bus powered with an auxiliary power source.

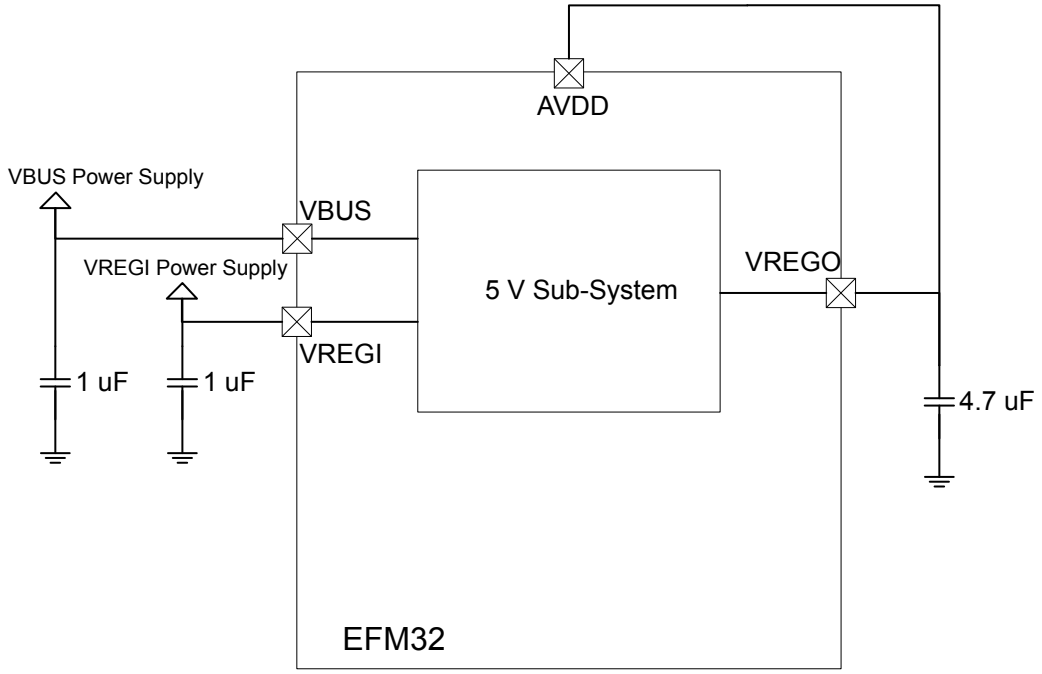


Figure 9.7. 5 V Sub-System Powered Device Configuration With VREGI and VBUS Connected to Source Supply

Figure 9.8 5 V Sub-System Powered Device Configuration With Only VBUS Connected to Source Supply on page 290 shows the recommended configuration for powering up a device using the 5 V sub-system with only the VBUS input connected to a source supply. This type of configuration can also be used for an application that is fully USB bus-powered.

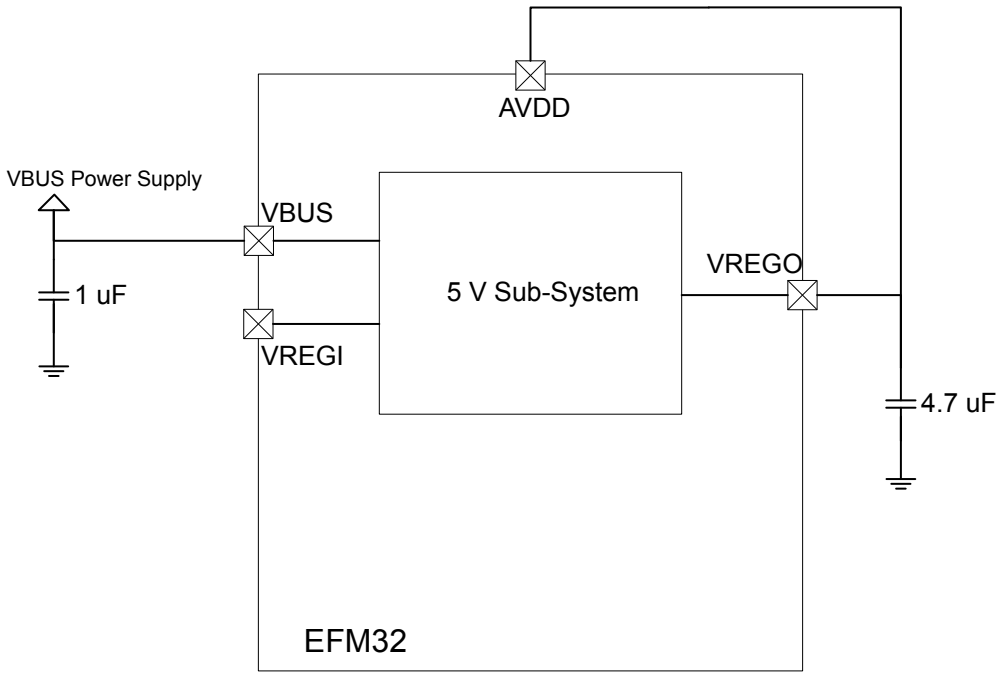


Figure 9.8. 5 V Sub-System Powered Device Configuration With Only VBUS Connected to Source Supply

Figure 9.9 5 V Sub-System Powered Device Configuration With Only VREGI Connected to Source Supply on page 290 shows the recommended configuration for powering up a device using the 5 V sub-system with only the VREGI input connected to a source supply.

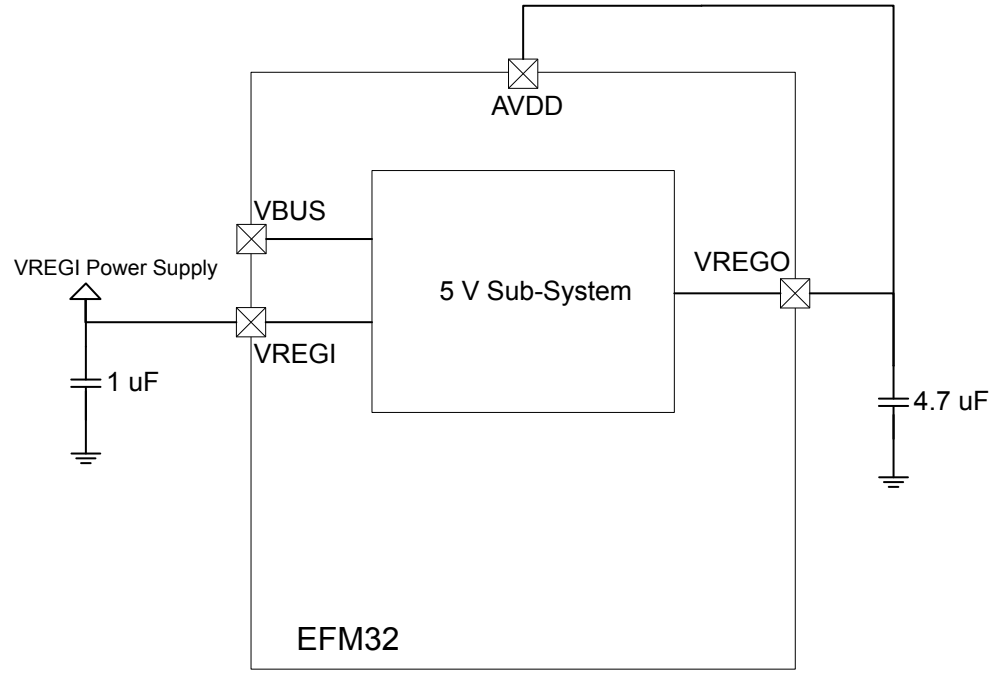


Figure 9.9. 5 V Sub-System Powered Device Configuration With Only VREGI Connected to Source Supply

9.3.14.1.2 USB Self-Powered Device

Although in most systems the 5 V sub-system provides power and AVDD is externally shorted to VREGO, in the case of a USB self-powered device, only the USB PHY is powered by the 5 V sub-system and AVDD is powered from another voltage source. In this case, the 5 V regulator does not start until both the 3 V and 5 V power supplies are present. The output VREGO can be used to power up external devices. The EMU_R5 VCTRL_INPUTMODE should be set to VREGI in this configuration so that the USB self-powered device derives power only from VREGI.

Figure 9.10 USB Self-Powered Device Configuration With VREGI and VBUS Connected to Source Supply on page 291 shows the recommended USB self-powered device configuration with both VREGI and VBUS inputs connected to source supplies. In this configuration, the regulator draws current from VREGI and the VBUS line is used only to indicate the presence of USB.

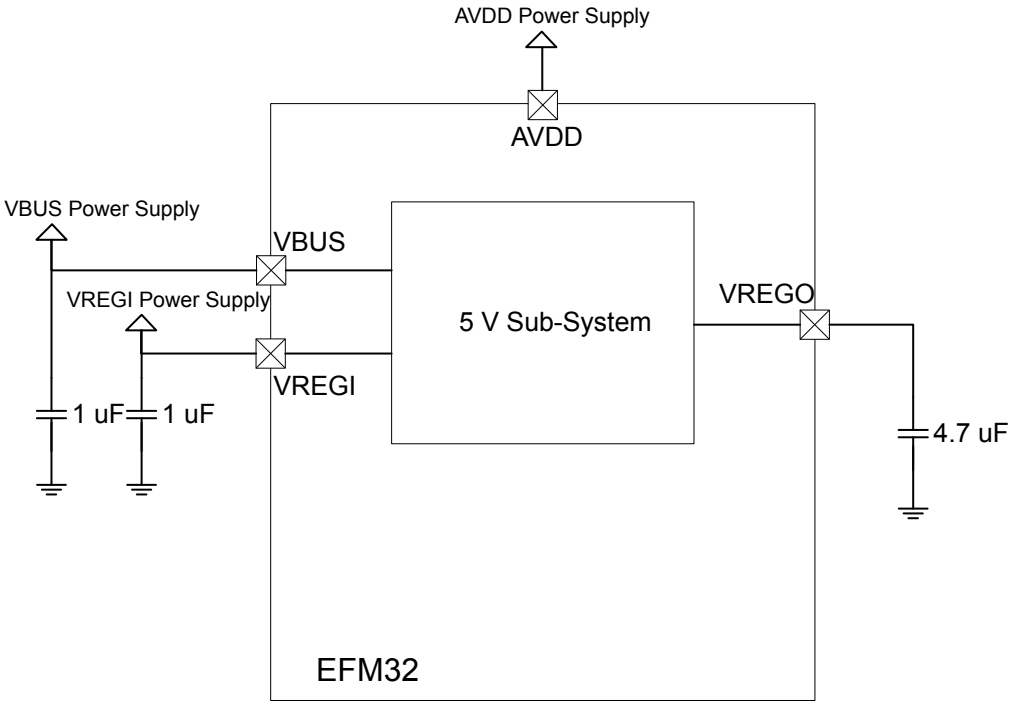


Figure 9.10. USB Self-Powered Device Configuration With VREGI and VBUS Connected to Source Supply

Figure 9.11 USB Self-Powered Device Configuration With Only VREGI Connected to Source Supply on page 292 shows the recommended USB self-powered device configurations with only VREGI connected to a source supply.

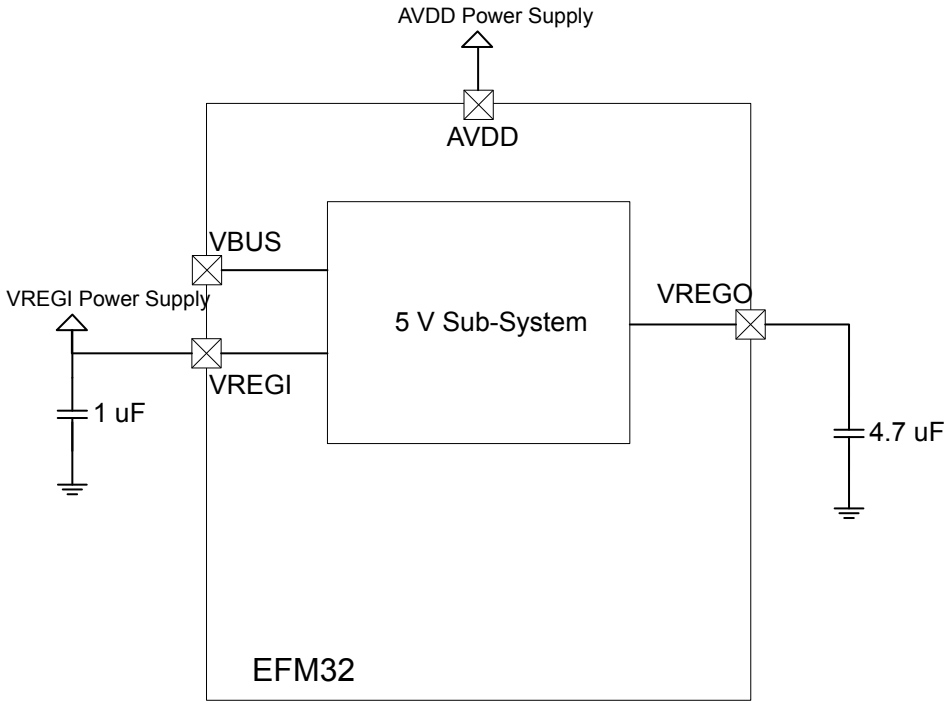


Figure 9.11. USB Self-Powered Device Configuration With Only VREGI Connected to Source Supply

9.3.14.1.3 Smartphone Powered Device

In certain smartphone accessory applications where a smartphone is the USB host, the power supply provided by the smartphone is may be less than the 5 V USB standard (typically the smartphone supply will be a nominal 3.3 V). In this case, VREGO may be driven externally from the smartphone power source rather than the 5 V sub-system. In typical use cases, AVDD is also powered from the same supply. In order to maintain the standard USB PHY operation, VBUS should be shorted to VREGO. The recommended configuration for such a smartphone accessory application is shown in Figure 9.12 Smartphone Powered Device Configuration on page 293.

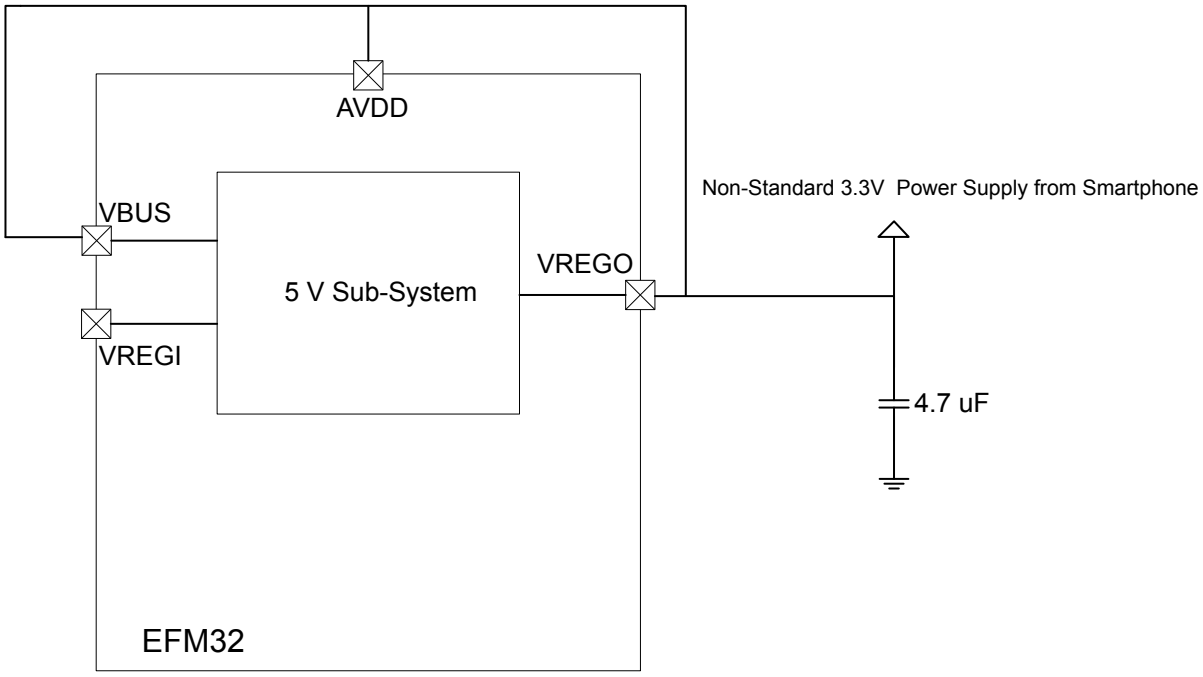


Figure 9.12. Smartphone Powered Device Configuration

9.3.14.2 Input and Output Configuration

The 5 V sub-system provides a general means of taking power from one of its two inputs VBUS and VREGI and generating a programmable output supply at VREGO. Figure 9.13 Input and Output Configuration Options on page 294 shows the internal connections of the 5 V sub-system inputs and output which can be configured by software via the EMU_R5VCTRL register. By default, the 5 V sub-system derives power from the higher of VBUS and VREGI voltages and the output VREGO is connected to the 5 V regulator output.

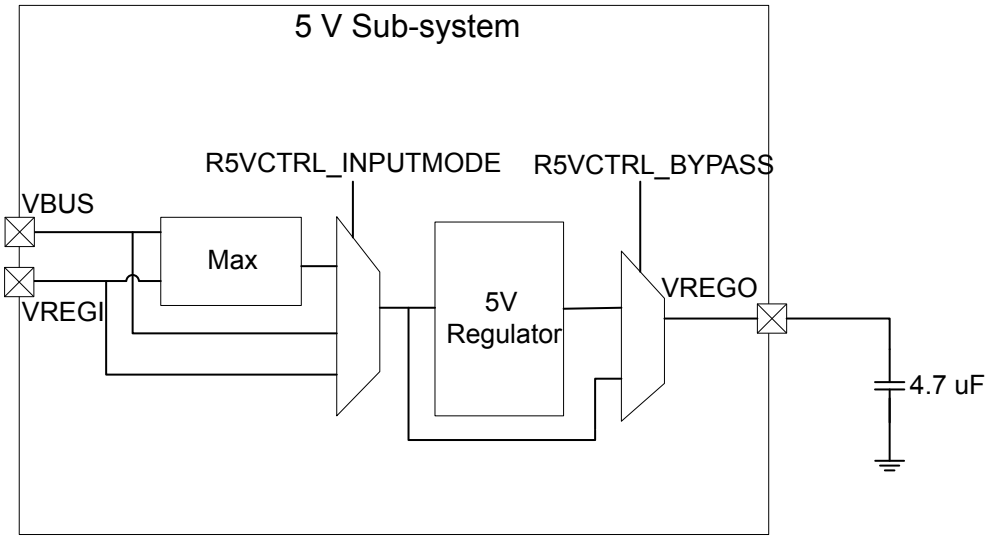


Figure 9.13. Input and Output Configuration Options

9.3.14.2.1 Input Modes

If the 5 V sub-system is used to power the device, during initial power up the 5 V sub-system always takes power from the higher of VBUS and VREGI. After initial power-up of the device, the EMU_R5VCTRL_INPUTMODE bitfield can be used to select which input(s) are used to provide power to the 5 V regulator.

When EMU_R5VCTRL_INPUTMODE is set to AUTO, the 5 V sub-system supply switches automatically to the higher voltage between VBUS and VREGI. When one of the input voltages falls below the voltage of the other input, this input setting will automatically select the higher input voltage as the input source of the 5 V sub-system. The 5 V sub-system can also be forced to take power only from VBUS or only from VREGI by setting EMU_R5VCTRL_INPUTMODE = VBUS or EMU_R5VCTRL_INPUTMODE = VREGI respectively. Note that EMU_R5VCTRL_INPUTMODE should be set to VREGI for USB self-powered devices to prevent power draw from the VBUS input.

9.3.14.2.2 Output Configuration

The output of the 5 V sub-system can be driven either from the output of the 5 V regulator (5 V Regulator Control Mode) or directly from the input (Bypass Mode).

9.3.14.2.2.1 5 V Regulator Control Mode

By default, the output of the 5 V sub-system is driven from the output of the 5 V regulator. When the output of the 5 V regulator controls the output of the 5 V sub-system, after the initial power up of the device the output voltage of the regulator can be programmed via EMU_R5VOUTLEVEL. The output voltage level is programmable in 100 mV increments between 2.4V to 3.8V depending on the value in the EMU_R5VOUTLEVEL_OUTLEVEL bitfield. EMU_R5VOUTLEVEL_OUTLEVEL should not be set to zero as the 5 V regulator does not support it. The EMU_R5VSYNC_OUTLEVELBUSY indicates the status of EMU_R5VOUTLEVEL and software must not re-write the EMU_R5VOUTLEVEL register until EMU_R5VSYNC_OUTLEVELBUSY reads 0. When software changes the regulator's output level using EMU_R5VOUTLEVEL_OUTLEVEL, it should wait for the EMU_IF_R5VVSINT before going into EM2 DeepSleep/EM3 Stop/EM4 Hibernate/Shutoff as the voltage update can only complete during EM0 Active/EM1 Sleep. The status bit EMU_STATUS_LDODROPOUTDET indicates whether the 5 V regulator input voltage is sufficient for the target output voltage. When the selected input to the 5 V regulator is insufficient to reach the programmed output level, the EMU_STATUS_LDODROPOUTDET status bit goes high. The example code below shows how to configure the regulator to produce a 3.3 V output voltage at VREGO.

```
/* inside EMU interrupt handler */
if(EMU->IF & EMU_IF_R5VVSINT)
   r5vvsint = 1;


/* inside main() */
r5vvsint = 0;
EMU->IEN |= EMU_IEN_R5VVSINT;
while(EMU->R5VSYNC & EMU_R5VSYNC_OUTLEVELBUSY) ;
EMU->R5VOUTLEVEL = (10 << _EMU_R5VOUTLEVEL_OUTLEVEL_SHIFT);
while (!r5vvsint) __WFI();
```

Figure 9.14 Programming 5 V Regulator Output on page 295 shows how the 5 V regulator output is updated to 3.3 V when software changes EMU_R5VOUTLEVEL_OUTLEVEL from 0x1 to 0xA. Each output voltage step is 100 mV and the wait time for each voltage step update is between 10 us and 15 us depending on the operating condition. The EMU_IF_R5VVSINT interrupt flag is set when the regulator has reached the desired output level. Note that the VREGO voltage waveform in Figure 9.14 Programming 5 V Regulator Output on page 295 corresponds to a 4.7 uF capacitor on VREGO as recommended.

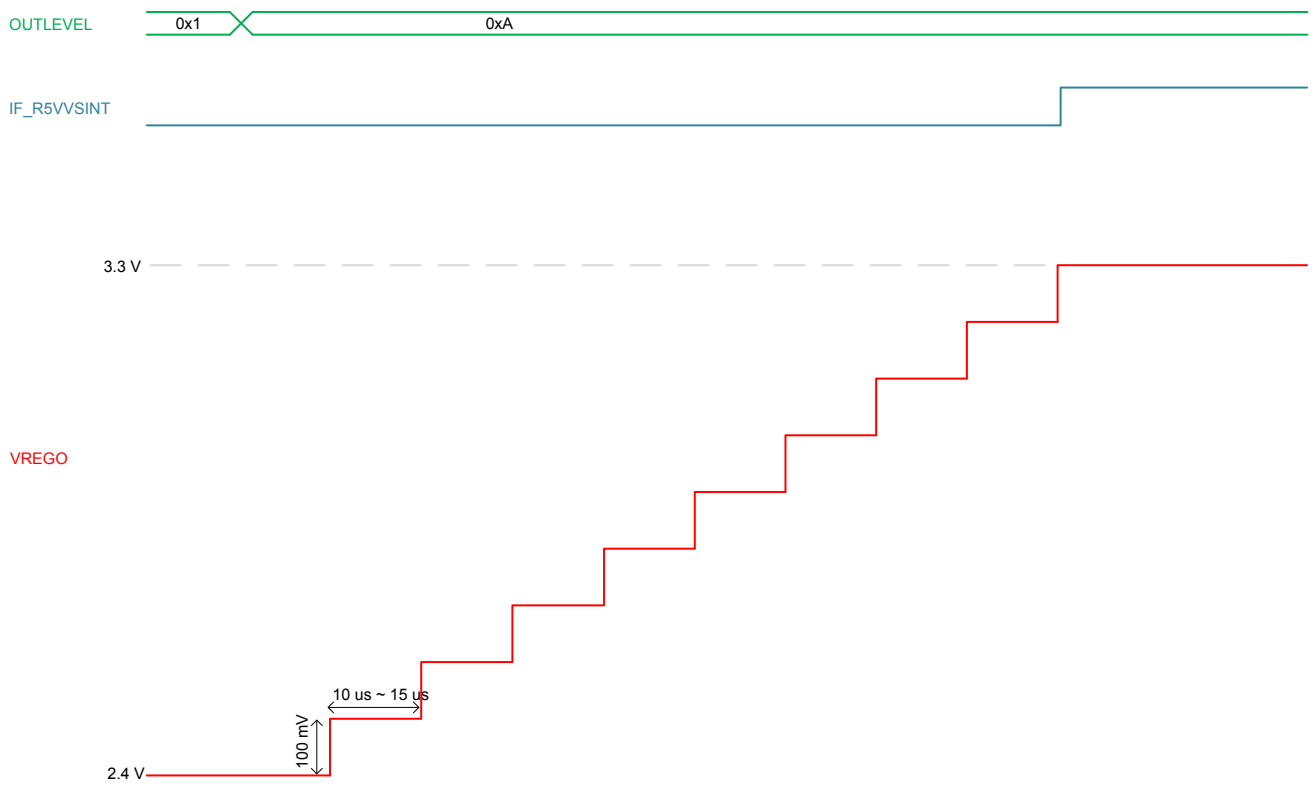


Figure 9.14. Programming 5 V Regulator Output

In order to maintain the programmed voltage at VREGO, at least one of the input supplies (VREGI and VBUS) needs to be present and should have sufficient voltage level to achieve the programmed output voltage level at VREGO. When both of the input supplies are disconnected or their voltage levels are very low and afterward one of the input power restoration happens while the system is in EM0 Active/EM1 Sleep, the output voltage at VREGO will be updated to the value programmed by EMU_R5VOUTLEVEL_OUTLEVEL. If the input power restoration happens when the system is in EM2 DeepSleep or EM3 Stop, the output voltage update will happen when the system returns to EM0 Active/EM1 Sleep. When the output voltage is being updated, software should not re-write the EMU_R5VOUTLEVEL register until EMU_R5VSYNC_OUTLEVELBUSY goes down. Software should also wait for EMU_IF_R5VVSINT interrupt before going into EM2 DeepSleep / EM3 Stop / EM4 Hibernate/Shutoff because the voltage update can only complete in EM0 Active/EM1 Sleep.

9.3.14.2.2.2 Bypass Mode

Instead of driving the 5 V sub-system output from the programmable output of 5 V regulator, the 5 V regulator can be bypassed and the 5 V sub-system output can be driven directly from the selected input by setting EMU_R5VCTRL_BYPASS to 1. Note that EMU_R5VCTRL_INPUTMODE must be set to VBUS or VREGI before setting EMU_R5VCTRL_BYPASS to 1. Bypass mode must not be used if the selected input voltage is above 3.8V or significant device damage may result. There are no voltage or current controls in bypass mode, so bypass mode cannot be used to charge the VREGO capacitor. Bypass mode must be used after the VREGO capacitor has been charged up to the desired output level. For example, if the selected 5 V sub-system input voltage is 3.3 V and 3.3 V is also desired at VREGO, the software must first update the output voltage to 3.3 V in 5 V regulator control mode as described in 9.3.14.2.2.1 5 V Regulator Control Mode. This voltage update will charge the VREGO capacitor as close to the desired output level as possible. Once the output voltage update is done, software can enable the bypass mode by setting EMU_R5VCTRL_BYPASS = 1.

The bypass mode is used when no regulation is required and it is desirable to save power by disabling the regulator. One example use case is when a system connects a Li-Ion battery to VREGI and EMU_R5VCTRL_INPUTMODE is set as VREGI to derive power supply from VREGI, and due to discharging the VREGI voltage falls too low to regulate the 5 V regulator output to the desired voltage level. The bypass mode can be useful for extending the useful battery life in this case. When the 5 V sub-system output is configured to be controlled by the 5 V regulator output, the status of the 5 V regulator input can be observed by polling the status bit EMU_STATUS_LDODROPOUTDET. The status bit EMU_STATUS_LDODROPOUTDET goes high as soon as the selected input voltage is not adequate to keep the output voltage at the desired level. When EMU_STATUS_LDODROPOUTDET goes high, software can enable the bypass mode to prolong battery life. Note that bypass mode can result in a voltage drop of up to 300 mV from the input to the output, depending on the amount of current being sourced. For applications such as certain smartphone accessory applications, where the smartphone is the USB Host and provides a non-standard power supply, it is recommended to power VREGO directly from the external supply instead of relying on bypass mode.

9.3.14.3 Energy Modes

The 5 V sub-system is available in all energy modes from EM0 Active down to EM4 Shutoff. However, when the 5 V sub-system is used, the following two conditions must be satisfied before making any energy mode transition from EM0 Active/EM1 Sleep.

1. On startup, software should enable EMU_IF_R5VREADY by setting EMU_IEN_R5VREADY = 1 and wait for the EMU_IF_R5VREADY interrupt before making any energy mode transition from EM0 Active/EM1 Sleep. The EMU_IF_R5VREADY interrupt flag indicates that the 5 V sub-system is ready for use. Apart from the EMU_IF_R5VREADY interrupt, the status of the 5 V sub-system can also be observed by polling EMU_R5VSTATUS_COLDSTART, which stays high when the 5 V sub-system is going through a cold start process and goes low when 5 V sub-system's cold start ends and it is ready for use. Any energy mode transition from EM0 Active/EM1 Sleep is allowed only when the 5 V sub-system is ready. The example code below shows how to make sure the 5 V sub-system is ready using the EMU_IF_R5VREADY interrupt flag before making any energy mode transition from EM0 Active/EM1 Sleep after startup.

```
/* inside EMU interrupt handler */
if(EMU->IF & EMU_IF_R5VREADY)
   r5ready = 1;


 /* inside main() */
 r5ready = 0;
 EMU->IEN |= EMU_IEN_R5VREADY;
 while (!r5ready) __WFI();
 <code for going into EM2/EM3/EM4>
```

2. When software changes the 5 V regulator's output level using EMU_R5VOUTLEVEL_OUTLEVEL, it should enable EMU_IF_R5VVSINT by setting EMU_IEN_R5VVSINT = 1 and wait for the EMU_IF_R5VVSINT before making any energy mode transition. The example code below shows how to make sure the 5 V sub-system output voltage update is done using the EMU_IF_R5VVSINT interrupt flag before making any energy mode transition from EM0 Active/EM1 Sleep.

```
/* inside EMU interrupt handler */
if(EMU->IF & EMU_IF_R5VVSINT)
   r5vvsint = 1;


/* inside main() */
r5vvsint = 0;
EMU->IEN |= EMU_IEN_R5VVSINT;
while(EMU->R5VSYNC & EMU_R5VSYNC_OUTLEVELBUSY) ;
EMU->R5VOUTLEVEL = (10 << _EMU_R5VOUTLEVEL_OUTLEVEL_SHIFT);
while (!r5vvsint) __WFI();
<code for going into EM2/EM3/EM4>
```

Table 9.7 Output Voltage and Current Sourcing Capability in Different Output Configurations With Respect to Different Energy Modes on page 297 shows the 5 V sub-system output voltage and current sourcing capability in different output configurations with respect to different energy modes.

Table 9.7. Output Voltage and Current Sourcing Capability in Different Output Configurations With Respect to Different Energy Modes

Energy Mode	Output Configuration			
	5 V Regulator Control Mode		Bypass Mode	
	Output Voltage	Current Sourcing Capability	Output Voltage	Current Sourcing Capability
EM0 Active, EM1 Sleep	Determined by the EMU_R5VOUTLEVEL_OUTLEVEL	Up to 200 mA with a maximum dropout of 300 mV	Selected input voltage (VBUS or VREGI) with a maximum dropout of 300 mV	Up to 200 mA with a maximum dropout of 300 mV

Energy Mode	Output Configuration			
	5 V Regulator Control Mode		Bypass Mode	
	Output Voltage	Current Sourcing Capability	Output Voltage	Current Sourcing Capability
EM2 DeepSleep, EM3 Stop, EM4 Hibernate	Determined by the EMU_R5VOUTLEVEL_OUTLEVEL	Up to 2 mA with a maximum dropout of 300 mV	Selected input voltage (VBUS or VREGI) with a maximum dropout of 300 mV	Up to 200 mA with a maximum dropout of 300 mV
EM4 Shutoff	Roughly the minimum of voltage of the selected input (VBUS or VREGI) and 3 V [1]	Up to 20 uA	Selected input voltage (VBUS or VREGI) with a maximum dropout of 300 mV	Up to 200 mA with a maximum dropout of 300 mV

Note:

1. The output voltage level can vary with process, temperature and load, and so it should not be used if a precision voltage is needed.

9.3.14.4 Supply Detection

The 5 V sub-system contains individual supply detectors on each of the VREGO, VBUS and VREGI pins. The three status bits EMU_R5VSTATUS_VREGODET, EMU_R5VSTATUS_VBUSDET and EMU_R5VSTATUS_VREGIDET indicate the presence of the voltage on VREGO, VBUS and VREGI respectively. Note that, the VREGO detector works only when VBUS and/or VREGI is also powered. See the device data sheet for detailed specifications regarding the detection threshold and hysteresis of these detectors. The voltage detectors for VREGO, VBUS and VREGI can be disabled individually by writing 1 to EMU_R5VDETCTRL_VREGODETDIS, EMU_R5VDETCTRL_VBUSDETDIS and EMU_R5VDETCTRL_VREGIDETDIS respectively and can be re-enabled by writing 0 to the same bitfield.

9.3.14.4.1 Waking Up on VBUS Detect

The USB interrupt flags USB_IF_VBUSDETH and USB_IF_VBUSDETL reflect the transitions of the output of supply detector on VBUS. The USB_IF_VBUSDETH interrupt flag is set when the VBUS detector detects the presence of a voltage greater than or equal to a certain threshold at VBUS pin whereas the USB_IF_VBUSDETL interrupt flag becomes high when the VBUS voltage goes lower than the threshold. See the device data sheet for the detailed specification of the threshold value. The VBUSDETH and VBUSDETL interrupts can be enabled by setting USB_IEN_VBUSDETH and USB_IEN_VBUSDETL high respectively. Both of these interrupts can wake the device up from EM2 DeepSleep when enabled. See 38.6 Register Map for detailed descriptions of the registers related to VBUSDETH and VBUSDETL interrupts.

Only the VBUSDETH interrupt can wake the system up from EM4 Hibernate/Shutoff. For waking up from EM4 Hibernate/Shutoff on the USB_IF_VBUSDETH interrupt, the USB_IEN_VBUSDETH as well as EMU_R5VCTRL_EM4WUEN must be set. The example below shows the minimum settings required for using USB_IF_VBUSDETH interrupt to wake the device up from EM4 Hibernate/Shutoff on VBUS detection.

```
USB->IEN |= USB_IEN_VBUSDETH;
EMU->R5VCTRL |= EMU_R5VCTRL_EM4WUEN;
```

9.3.14.5 Supply Comparison and Monitoring

The 5 V sub-system contains a supply comparator that compares between supplies connected to VBUS and VREGI input pins. The EMU_STATUS_VBUSGTVREGI shows the output of the comparator. When EMU_STATUS_VBUSGTVREGI status bit is 1, it indicates that the VBUS supply has a higher voltage than the VREGI supply. The opposite (VBUS voltage < VREGI voltage) is true when the status bit is 0.

The 5 V sub-system provides an option to monitor the voltage on VREGO, VBUS and VREGI and also to monitor the current through VBUS and VREGI by using the ADC. In order to meet the ADC's input voltage specifications, the VREGO voltage is scaled down by a factor of 6 and the VBUS and VREGI voltages are scaled down by a factor of 10. The currents through VBUS and VREGI are measured by converting the current to a voltage and measuring that voltage via the ADC. The nominal value of the voltage-to-current transfer ratio is 2.79. The current monitor circuitry is calibrated in production for better current measurement accuracy in the application. The CURRMON5V value in the DI page (see 4.6 DI Page Entry Map) stores this calibration parameter, which can be used to calculate the current fthat an ADC reading represents.

Only one of the two input pins and either voltage or current of that pin can be monitored at a time. The 5 V sub-system multiplexer selector EMU_R5VADCCTRL_AMUXCEL controls which of these values will be measured by the ADC. In order to enable this 5 V sub-system multiplexer, EMU_R5VADCCTRL_ENAMUX needs to be set to 1. When monitoring current, the EMU_R5VCTRL_IMONEN is also required to be set to 1.

The ADC should be configured to operate in single channel mode. The 5 V reference to ADC single channel mode should be selected and the ADC single channel mode positive and negative input should select R5VOUT and VSS respectively. Example code for setting up the ADC0 to monitor the VREGO output is shown below.

```
ADC0->SINGLECTRL = (ADC_SINGLECTRL_REF_5V |
                    ADC_SINGLECTRL_POSSEL_R5VOUT |
                    ADC_SINGLECTRL_NEGSEL_VSS);
EMU->R5VADCCTRL = EMU_R5VADCCTRL_AMUXSEL_VREGODIV6 | EMU_R5VADCCTRL_ENAMUX;
```

See 28.3.17 ADC Programming Model for details about the ADC programming model.

The output of the 5 V sub-system multiplexer is also available to the ACMP and can be selected by setting ACMPn_INPUTSEL_POSSEL = R5VOUT. See 27.3.6 Input Selection for details.

Every ADC and ACMP module in a device can use the 5 V sub-system multiplexer output as input.

9.3.15 Temperature Sensor

EMU provides low energy periodic temperature measurement. A temperature measurement is taken every 250 ms, with the 8-bit result stored in EMU->TEMP register.

Note: The EMU temperature sensor is always running (except in EM4 Shutoff) and is independent from the ADC temperature sensor.

The EMU provides the following features around temperature changes

- Wake-Up from EM4 Hibernate on Temperature Change
- Interrupt from High Level Trip
- Interrupt from Low Level Trip

During production test, the EMU temperature sensor for each device is calibrated at room temperature, with the corresponding calibration temperature and reading stored off in the DI page as follows:

- DEVINFO->CAL.TEMP : This bitfield contains the temperature in degrees C at calibration
- DEVINFO->EMUTEMP : This register contains the EMU->TEMP reading at the calibration temperature stored in DEVINFO->CAL.TEMP

The current calibrated EMU temperature sensor result from EMU->TEMP may be converted to degrees C using the following equation:

$$T_{J\_EMU}\,[°C] = (\,DEVINFO\text{->}CAL.TEMP\,) + (TEMPCO_{EMxx}) * [\,(DEVINFO\text{->}EMUTEMP) - (EMU\text{->}TEMP)\,]$$

Figure 9.15. Temperature Calculation

$TEMPCO_{EMxx}$ is a temperature coefficient that varies based on the energy mode at the time of the EMU temperature sensor reading:

- $TEMPCO_{EM01}$ = 0.278 + (DEVINFO->EMUTEMP) / 100
- $TEMPCO_{EM234}$ = 0.268 + (DEVINFO->EMUTEMP) / 100

For maximum accuracy when using the high/low level temperature interrupts, firmware should ensure that $TEMPCO_{EM234}$ is used to set the temperature thresholds in EMU->TEMPLIMITS before entering EM2/3/4. Similarly, when exiting EM2/3/4, the temperature thresholds should be updated using $TEMPCO_{EM01}$.

Note that an increasing reading in EMU->TEMP corresponds to a decreasing temperature, and vice-versa. If enabled, the TEMPHIGH High Level Limit in EMU-> TEMPLIMITS causes an interrupt flag on a increasing EMU->TEMP reading (i.e., decreasing temperature). Similarly, the TEMPLOW Low Level Limit causes a interrupt flag on a decreasing EMU->TEMP reading (i.e., increasing temperature).

The EMU temperature sensor accuracy is approximately ±10°C over most of the useable temperature range, but may be +15°C at higher temperatures. Accordingly, any use of the EMU temperature sensor should include margin to account for that accuracy.

9.3.16 Registers latched in EM4

The following registers will be latched when entering EM4. After wake-up from EM4, these registers will be reset and require reprogramming prior to writing the EMU_CMD_EM4UNLATCH command.

- CMU_LFRCOCTRL
- CMU_LFXOCTRL
- CMU_LFECLKSEL
- CMU_LFECLKEN0
- CMU_LFEPRESC0

9.3.17 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.

9.3.18 Backup Power Domain

EFM32 Giant Gecko 12 has the possibility to be partly powered by backup battery. The backup power input, BU_VIN, is connected to a power domain in the EFM32 Giant Gecko 12 containing the RTCC, 128 bytes of data retention and the CRYOTIMER. Figure 9.16 Backup Power Domain Overview on page 301 shows an overview of the backup powering scheme. During normal operation, the entire chip is powered by the main power supply. If the main power supply drains out and the backup mode functionality is enabled, the system enters a low energy mode, equivalent to EM4 Hibernate, and automatically switches over to the backup power supply. The power relationship requirements given in the 9.3.4 Power Configurations must always be adhered to. This means that even when the main supply (VREGVDD/AVDD) falls, the power relationships must stay valid (I.e., IOVDD must be less than or equal to AVDD in all scenarios.

Note: If there is no backup battery inserted in the system, then DISMAXCOMP in EMU_BUCTRL can be set to 1 to save power. This bit must be set to 0 (default value) if the backup battery is present in the system (even if backup mode is not enabled).

Consult the data sheet for the allowable BU_VIN input voltage range.

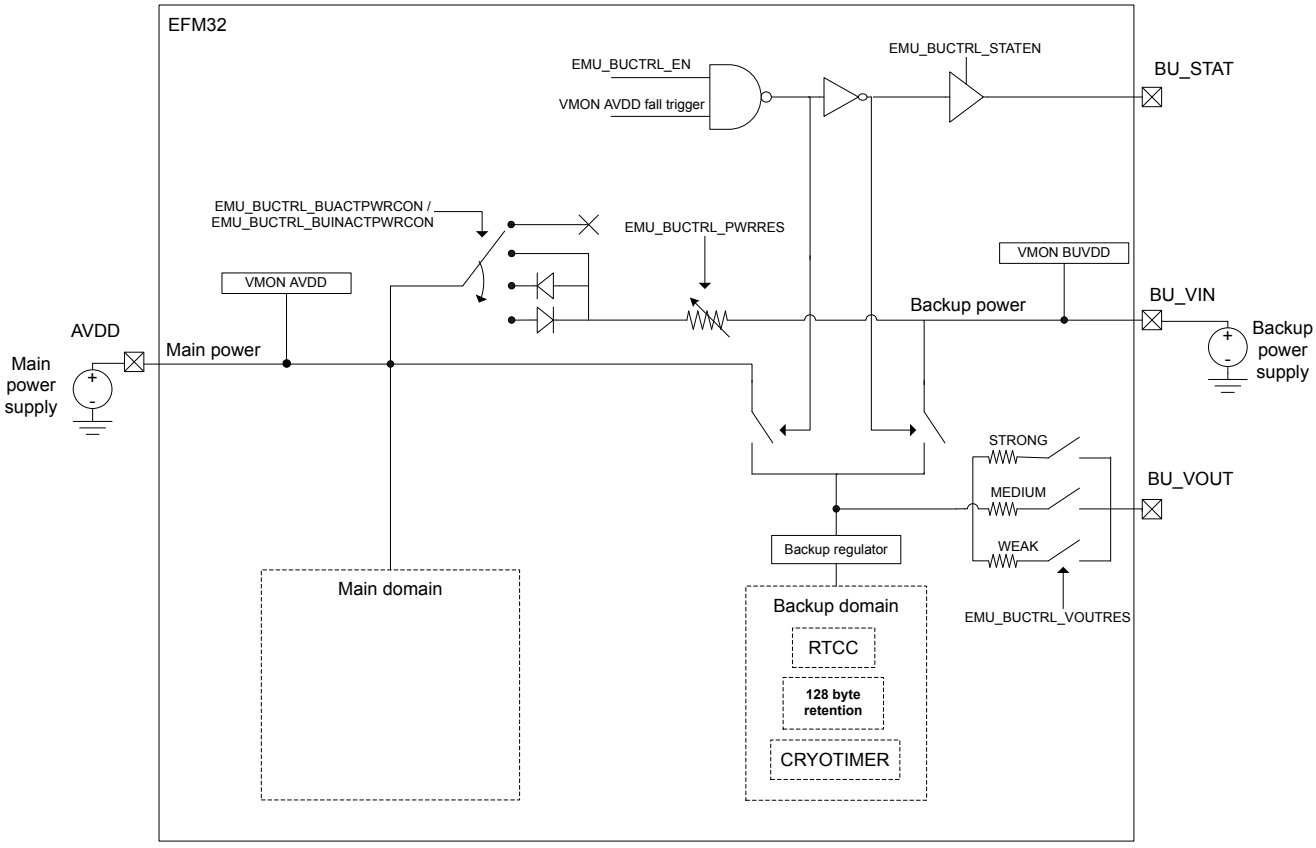


Figure 9.16. Backup Power Domain Overview

9.3.18.1 Entering Backup Mode

To be able to enter backup mode, the EN bit in EMU_BUCTRL and the EN bit in EMU_VMONAVDDCTRL have to be set. When these two are set, the BURDY interrupt flag will be set as soon as the VMON AVDD channel is ready. Status of the backup functionality is also available in the BURDY flag in the EMU_STATUS register. To enter backup mode, the voltage on AVDD has to drop below the programmed fall threshold of the VMON AVDD channel. This threshold is programmed using FALLTHRESCOARSE and FALLTHRESFINE in EMU_VMONAVDDCTRL.

In the above mentioned case, EFM32 Giant Gecko 12 will try to enter backup mode even if there is no backup power supply present. If the EN bit in EMU_VMONBUVDDCTRL is set as well and AVDD falls below fall threshold, then backup mode will only be entered if BUVDD is above a programmed threshold. This threshold can be programmed using THRESCOARSE and THRESFINE in the EMU_VMONBUVDDCTRL. BURDY status flag will go high when the EN bit in EMU_BUCTRL and the EN bit in EMU_VMONAVDDCTRL are set. If the EN bit in EMU_VMONBUVDDCTRL is now set, then immediately after that the user should wait on the BURDY status flag to first go low and then go high again (the flag will go high when the VMON BUVDD has also become ready). Note that enabling the BUVDD VMON monitoring causes a drain of 2 µA from the backup power supply in EM0 Active.

The BU_STAT pin can be used to indicate whether or not the system is in backup mode. To enable exporting of the backup mode status to BU_STAT pin, set STATEN in EMU_BUCTRL. When enabled, BU_STAT pin is driven to BU_VIN if backup mode is active and to ground otherwise.

Note:

- When EN in EMU_BUCTRL is set, then EM4 Shutoff entry is blocked. All software based EM4 Shutoff entries will result in an entry to EM4 Hibernate. This is a safety feature since it is not possible to enter backup mode if the chip is in EM4 Shutoff.
- If DCDC is ON, it is turned off when entering backup mode.
- The RTCC includes functionality for storing a timestamp when the system enters backup mode. See the RTCC chapter for details.

9.3.18.2 Exiting Backup Mode

To exit backup mode, the voltage on AVDD has to be above the rise threshold programmed in EMU_VMONAVDDCTRL. RISETHRECOARSE and RISETHRESFINE in EMU_VMONAVDDCTRL decides threshold for backup mode exit. When leaving backup mode, a system reset is triggered (same as EM4 Hibernate exit) in which backup domain is not reset. When backup mode has been active, the BUMODERST bit in RMU_RSTCAUSE is set (both EM4RST and BUMODERST bits in RMU_RSTCAUSE will be set if checked after backup mode exit). Figure 9.17 Entering and Leaving Backup Mode on page 302 illustrates how the VMON monitoring on AVDD can be programmed to implement hysteresis on entering and exiting backup mode.

Backup mode is also exited on a hard pin reset or if a brown out occurs on the backup power supply.

Note: Exit from backup mode on AVDD rise happens independent of whether the RISEWU bit in EMU_VMONAVDDCTRL is set or not.

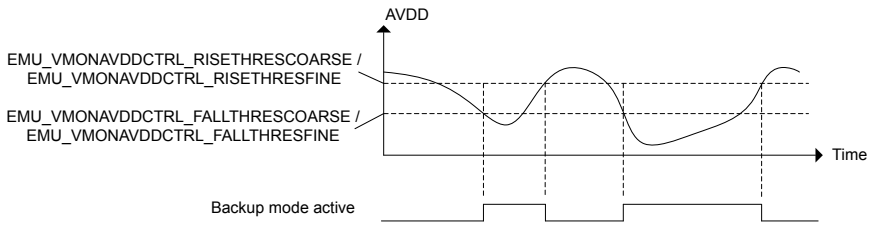


Figure 9.17. Entering and Leaving Backup Mode

9.3.18.3 Backup Pads

There are three backup pads, i.e., BU_VIN, BU_VOUT and BU_STAT. When these are being used for backup, then no other module should drive these pads. Following sequence can be followed to ensure this:

- These should be disabled using the Mode register (GPIO_Px_MODEL/GPIO_Px_MODEH)
- DOUT for these should be set to 0 using GPIO_Px_DOUT (if DOUT remains set, then these will be pulled-up even if disabled)
- These should be locked using GPIO_Px_PINLOCKN

9.3.18.4 Threshold Calibration

Thresholds for backup entry and backup exit are monitored by VMON. Calibrated threshold values (coarse, fine) for two voltages are given in DI page for all VMON channels. All other values between these two voltages can be found by linear interpolation.

9.3.18.5 Backup Battery Charging

The EFM32 Giant Gecko 12 includes functionality for charging of the backup battery. This is done by connecting the main power and the backup power through a resistor, and optionally a diode. The connection is configured individually for when in backup mode and when in normal mode. When in normal mode, the connection is configured in BUINACTPWRCON in EMU_BUCTRL. BUACTPWRCON in EMU_BUCTRL configures the connection when in backup mode. The series resistance between the two power domains is configured in PWRRES in EMU_BUCTRL, this configuration applies both to backup mode and normal mode.

9.3.18.6 Supply Voltage Output

To be able to power external devices, the supply voltage for the backup domain is available as an output. Three switches connect the backup supply voltage to the BU_VOUT pin. To be able to control the series resistance, the switches have different strengths: weak, medium, and strong (strong connection has the lowest resistance). The switches are controlled using the VOUTRES in EMU_BUCTRL. For resistor values, refer to the device data sheet Electrical Characteristics.

9.3.18.7 Voltage Probing

It is possible to internally probe the voltage levels at AVDD and BU_VIN using the ADC. To probe AVDD, AVDD needs to be selected in POSSEL of ADCn_SINGLECTRL before performing the ADC conversion. In order to probe BU_VIN, BUVINPROBEEN in EMU_BUCTRL needs to be set first. Then BUVDD needs to be selected in the POSSEL of ADCn_SINGLECTRL before performing the conversion. The voltage measured by the ADC on the BUVDD channel will be 1/8 of the actual BU_VIN voltage, meaning that the result needs to be multiplied by 8 to get the correct measurement. BU_VOUT cannot be probed internally. However, BU_VOUT can be externally connected to any pin accessible by the ADC. When making the external connection, the user must ensure that the ADC pin is not driven to a voltage higher than the ADC power when the chip is in main mode (i.e., not in backup mode). This is already taken care of in the chip if the ADC is not powered by the DCDC (i.e., in main mode, BU_VOUT is the same as AVDD and AVDD also powers the ADC). If the DCDC is used to power the ADC, then the user must divide BU_VOUT outside the chip by at least a factor of 4 before feeding it to the ADC pin. In backup mode, external connection of BU_VOUT with the ADC pin does not create any issues (as the ADC is not available in backup mode).

9.3.18.8 EM4 Hibernate vs Backup Mode

Backup mode is a special version of EM4 Hibernate, with only three key differences:

- EM4 Hibernate has GPIO retention capability which is not present in backup mode.
- EM4 Hibernate has multiple wakeup sources shown in Table 9.3 EMU Wake-Up Triggers from Low Energy Modes on page 276. Backup mode can only be exited when AVDD goes above rise threshold, on a hard pin reset or on a brown out on BU_VIN.

9.3.18.9 Oscillators in Backup Mode

Backup mode is equivalent to EM4 Hibernate, therefore oscillators available in EM4 Hibernate are also available in backup mode. Note that the chosen oscillator may need to be retained (using EMU_EM4CTRL) in order to keep it running in backup mode (settings done for EM4 Hibernate also apply to backup mode).

9.3.18.10 Brown Out Detection in Backup Mode

When backup mode is entered, the EM4BOD switches from monitoring AVDD to monitoring BU_VIN. A brown out on BU_VIN (treated the same as AVDD brown out in EM4 Hibernate) results in an exit from backup mode.

9.4 Register Map

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

Offset	Name	Type	Description
0x000	EMU_CTRL	RW	Control Register
0x004	EMU_STATUS	R	Status Register
0x008	EMU_LOCK	RWH	Configuration Lock Register
0x00C	EMU_RAM0CTRL	RW	Memory Control Register
0x010	EMU_CMD	W1	Command Register
0x018	EMU_EM4CTRL	RW	EM4 Control Register
0x01C	EMU_TEMPLIMITS	RW	Temperature Limits for Interrupt Generation
0x020	EMU_TEMP	R	Value of Last Temperature Measurement
0x024	EMU_IF	R	Interrupt Flag Register
0x028	EMU_IFS	W1	Interrupt Flag Set Register
0x02C	EMU_IFC	(R)W1	Interrupt Flag Clear Register
0x030	EMU_IEN	RW	Interrupt Enable Register
0x034	EMU_PWRLOCK	RW	Regulator and Supply Lock Register
0x03C	EMU_PWRCTRL	RW	Power Control Register
0x040	EMU_DCDCCTRL	RW	DCDC Control
0x04C	EMU_DCDCMISCCTRL	RW	DCDC Miscellaneous Control Register
0x050	EMU_DCDCZDETCTRL	RW	DCDC Power Train NFET Zero Current Detector Control Register
0x054	EMU_DCDCCLIMCTRL	RW	DCDC Power Train PFET Current Limiter Control Register
0x058	EMU_DCDCLNCOMPCTRL	RW	DCDC Low Noise Compensator Control Register
0x05C	EMU_DCDCLNVCTRL	RWH	DCDC Low Noise Voltage Register
0x064	EMU_DCDCLPVCTRL	RW	DCDC Low Power Voltage Register
0x06C	EMU_DCDCLPCTRL	RW	DCDC Low Power Control Register
0x070	EMU_DCDCLNFREQCTRL	RW	DCDC Low Noise Controller Frequency Control
0x078	EMU_DCDCSYNC	R	DCDC Read Status Register
0x090	EMU_VMONAVDDCTRL	RW	VMON AVDD Channel Control
0x094	EMU_VMONALTAVDDCTRL	RW	Alternate VMON AVDD Channel Control
0x098	EMU_VMONDVDDCTRL	RW	VMON DVDD Channel Control
0x09C	EMU_VMONIO0CTRL	RW	VMON IOVDD0 Channel Control
0x0A0	EMU_VMONIO1CTRL	RW	VMON IOVDD1 Channel Control
0x0A4	EMU_VMONBUVDDCTRL	RW	VMON BUVDD Channel Control
0x0B4	EMU_RAM1CTRL	RW	Memory Control Register
0x0B8	EMU_RAM2CTRL	RW	Memory Control Register
0x0BC	EMU_BUCTRL	RW	Backup Power Configuration Register
0x0C8	EMU_R5VCTRL	RW	5V Regulator Control
0x0CC	EMU_R5VADCCTRL	RW	5V Regulator Control

Offset	Name	Type	Description
0x0D0	EMU_R5VOUTLEVEL	RW	5V Regulator Voltage Select
0x0DC	EMU_R5VDETCTRL	RW	5V Detector Enables
0x0EC	EMU_DCDCLPEM01CFG	RW	Configuration Bits for Low Power Mode to Be Applied During EM01, This Field is Only Relevant If LP Mode is Used in EM01
0x0F0	EMU_R5VSTATUS	R	5V Detector Status Register
0x0F8	EMU_R5VSYNC	R	5V Read Status Register
0x100	EMU_EM23PERNORETAINCMD	W1	Clears Corresponding Bits in EM23PERNORETAINSTATUS Unlocking Access to Peripheral
0x104	EMU_EM23PERNORETAINSTA-TUS	R	Status Indicating If Peripherals Were Powered Down in EM23, Subsequently Locking Access to It
0x108	EMU_EM23PERNORETAINCTRL	RW	When Set Corresponding Peripherals May Get Powered Down in EM23

9.5 Register Description

9.5.1 EMU_CTRL - 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								0x0					0	0	0	0	
Access															RW								RW					RW	RW	RW	RW	
Name															EM4HVSCALE								EM23VSCALE					EM23VSCALEAUTOWSEN	EM01LD	EM2BODDIS	EM2BLOCK	

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	EM4HVSCALE	0x0	RW	**EM4H Voltage Scale**

Set EM4H voltage. Entry to EM4H will trigger voltage scaling to this voltage if voltage scale level in EM4HVSCALE is less than that of VSCALE

Value	Mode	Description
0	VSCALE2	Voltage Scale Level 2
2	VSCALE0	Voltage Scale Level 0
3	RESV	RESV

15:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	EM23VSCALE	0x0	RW	**EM23 Voltage Scale**

Set EM23 voltage. Entry to EM2/3 will trigger voltage scaling to this voltage if voltage scale level in EM23VSCALE is lesser than that of VSCALE

Value	Mode	Description
0	VSCALE2	Voltage Scale Level 2
2	VSCALE0	Voltage Scale Level 0
3	RESV	RESV

7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	EM23VSCALEAUTOWSEN	0	RW	**Automatically Configures Flash and Frequency to Wakeup From EM2 or EM3 at Low Voltage**

With voltage scaling on EM2/3 entry, wakeup to EM0/1 will be at the same voltage as EM2. When this bit is set the Flash wait states and CMU clock frequency are automatically configured to safe value without needing software to configure it prior to EM2/3 entry.

Bit	Name	Reset	Access	Description
3	EM01LD	0	RW	**Reserved for internal use. Do not change.**
	Reserved for internal use. Do not change.			
2	EM2BODDIS	0	RW	**Disable BOD in EM2**
	This bit is used to disable BODs to minimize current in EM2. Reset with POR or Hard Pin Reset			
1	EM2BLOCK	0	RW	**Energy Mode 2 Block**
	This bit is used to prevent the MCU from entering Energy Mode 2 or 3.			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

9.5.2 EMU_STATUS - Status 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						0						0		0	0x0					0					0		0	0	0	0	0	0
Access						R						R		R	R					R					R		R	R	R	R	R	R
Name						TEMPACTIVE						EM4IORET		VSCALEBUSY	VSCALE					BURDY					VMONBUVDD		VMONIO1	VMONIO0	VMONDVDD	VMONALTAVDD	VMONAVDD	VMONRDY

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	TEMPACTIVE	0	R	**Temperature Measurement Active**
	This signal is set during EMU Temperature measurement			
25:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	EM4IORET	0	R	**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, and can also be cleared in EM4H by the VMON.			

Value	Mode	Description
0	DISABLED	IO retention is disabled.
1	ENABLED	IO retention is enbled.

19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	VSCALEBUSY	0	R	**System is Busy Scaling Voltage**
	Indicates that the system is busy scaling voltage in EM0/1. Writing EM01VSCALE0 or EM01VSCALE2 in EMU_CMD register while this is set will abort the current voltage scaling process and initiate a new one			
17:16	VSCALE	0x0	R	**Current Voltage Scale Value**
	This shows the current system voltage value. This gets updated after VSCALEDONE interrupt or on EM23 exit or EM4H exit			

Value	Mode	Description
0	VSCALE2	Voltage Scale Level 2
2	VSCALE0	Voltage Scale Level 0
3	RESV	RESV

15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	BURDY	0	R	**Backup Mode Ready**
	Set when the Backup power functionality is ready.			

Bit	Name	Reset	Access	Description
11:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	VMONBUVDD	0	R	**VMON BUVDD Channel**
	Indicates the status of the BUVDD channel of the VMON.			
6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	VMONIO1	0	R	**VMON IOVDD1 Channel**
	Indicates the status of the IOVDD1 channel of the VMON.			
4	VMONIO0	0	R	**VMON IOVDD0 Channel**
	Indicates the status of the IOVDD0 channel of the VMON.			
3	VMONDVDD	0	R	**VMON DVDD Channel**
	Indicates the status of the DVDD channel of the VMON.			
2	VMONALTAVDD	0	R	**Alternate VMON AVDD Channel**
	Indicates the status of the Alternate AVDD channel of the VMON.			
1	VMONAVDD	0	R	**VMON AVDD Channel**
	Indicates the status of the AVDD channel of the VMON.			
0	VMONRDY	0	R	**VMON Ready**
	VMON status. When high, this bit indicates that all the enabled channels are ready. When low, it indicates that one or more of the enabled channels are not ready.			

9.5.3 EMU_LOCK - Configuration Lock 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																	0x0000															
Access																	RWH															
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	0x0000	RWH	**Configuration Lock Key**

Write any other value than the unlock code to lock all EMU registers, except the interrupt registers and regulator control registers, from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.

Mode	Value	Description
Read Operation		
UNLOCKED	0	EMU registers are unlocked
LOCKED	1	EMU registers are locked
Write Operation		
LOCK	0	Lock EMU registers
UNLOCK	0xADE8	Unlock EMU registers

9.5.4 EMU_RAM0CTRL - Memory 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		
Access																														RW		
Name																														RAMPOWERDOWN		

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	RAMPOWERDOWN	0x0	RW	**RAM0 Blockset Power-down**

RAM blockset power-down in EM23 with full access in EM01. Block 0 may never be powered down.

Mode	Value	Description
NONE	0x00	None of the RAM blocks powered down
BLK3	0x4	Power down RAM block 3 (address range 0x2000C000-0x2001FFFF)
BLK2TO3	0x6	Power down RAM blocks 2 and above (address range 0x20008000-0x2001FFFF)
BLK1TO3	0x7	Power down RAM blocks 1 and above (address range 0x20004000-0x2001FFFF)

9.5.5 EMU_CMD - 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																										0		0				0
Access																										W1		W1				W1
Name																										EM01VSCALE2		EM01VSCALE0				EM4UNLATCH

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	EM01VSCALE2	0	W1	**EM01 Voltage Scale Command to Scale to Voltage Scale Level 2**
	Start EM01 voltage scaling to Voltage Scale Level 2. Write to this register will trigger voltage scaling to Voltage Scale Level 2 followed by an VSCALEDONE interrupt			
5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	EM01VSCALE0	0	W1	**EM01 Voltage Scale Command to Scale to Voltage Scale Level 0**
	Start EM01 voltage scaling to Voltage Scale Level 0. Write to this register will trigger voltage scaling to Voltage Scale Level 0 followed by an VSCALEDONE interrupt			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EM4UNLATCH	0	W1	**EM4 Unlatch**
	When entering EM4, several registers will be latched in order to maintain constant functionality throughout EM4. Upon wakeup, these registers will be reset and can have contradictory values to the latched values. To ensure a seamless transition from EM4 to EM0, the unlatch command should be given after properly reconfiguring these latched registers. The unlatch command can be executed after any reset condition but is only needed after EM4 wakeup.			

9.5.6 EMU_EM4CTRL - EM4 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												0x0		0	0	0	0
Access															W1												RW		RW	RW	RW	RW
Name															EM4ENTRY												EM4IORETMODE		RETAINULFRCO	RETAINLFXO	RETAINLFRCO	EM4STATE

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	EM4ENTRY	0x0	W1	**Energy Mode 4 Entry**
	This register 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.			
15:6	*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 Disable**
	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	RETAINULFRCO	0	RW	**ULFRCO Retain During EM4S**
	Retain the ULFRCO upon EM4S entry. If set to 1, an already running ULFRCO will be retained in its running state in EM4. ULFRCO will always be retained if EM4STATE is in EM4H.			
2	RETAINLFXO	0	RW	**LFXO Retain During EM4**
	Retain the LFXO upon EM4(SH/H) entry. If set to 1, an already running LFXO will be retained in its running state in EM4.			
1	RETAINLFRCO	0	RW	**LFRCO Retain During EM4**
	Retain the LFRCO upon EM4(S/H) entry. If set to 1, an already running LFRCO will be retained in its running state in EM4.			
0	EM4STATE	0	RW	**Energy Mode 4 State**
	When set, the system will enter Hibernate state (EM4H) when entering EM4. In EM4H, the regulator will be on in reduced mode allowing for RTCC. Otherwise, when entering in EM4, the regulator will be disabled allowing for lowest power mode, Shutoff state (EM4S). Only reset with POR or Hard Pin Reset			

Value	Mode	Description
0	EM4S	EM4S Shutoff state
1	EM4H	EM4H Hibernate state

9.5.7 EMU_TEMPLIMITS - Temperature Limits for Interrupt Generation

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																0	0xFF								0x00							
Access																RW	RW								RW							
Name																EM4WUEN	TEMPHIGH								TEMPLOW							

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	EM4WUEN	0	RW	**Enable EM4 Wakeup Due to Low/high Temperature**
	Enable EM4 wakeup from low or high temperature from EM4H			
15:8	TEMPHIGH	0xFF	RW	**Temperature High Limit**
	The TEMPHIGH interrupt flag is set when a periodic temperature measurement is equal to or higher than this value. If the high limit is changed during a temperature measurement (TEMPACTIVE=1), the limit update will be delayed until the end of the temperature measurement.			
7:0	TEMPLOW	0x00	RW	**Temperature Low Limit**
	The TEMPLOW interrupt flag is set when a periodic temperature measurement is equal to or lower than this value. If the low limit is changed during a temperature measurement (TEMPACTIVE=1), the limit update will be delayed until the end of the temperature measurement.			

9.5.8 EMU_TEMP - Value of Last Temperature Measurement

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																									0xXX							
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	0xXX	R	**Temperature Measurement**
	Value of last periodic temperature measurement. Value is asynchronously updated. Value is stable for 250 ms after a temperature-based interrupt (TEMPHIGH, TEMPLOW, or TEMP) and can be read with a single read operation. If register is read not in response to a temperature-based interrupt, multiple readings should be taken until two consecutive values are the same.			

9.5.9 EMU_IF - Interrupt Flag 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	0	0	0				0	0	0	0		0	0	0	0	0			0	0		0	0	0	0	0	0	0	0	0	0	0
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
Name	TEMPHIGH	TEMPLOW	TEMP				VSCALEDONE	EM23WAKEUP	R5VVSINT	BURDY		DCDCINBYPASS	DCDCLNRUNNING	DCDCLPRUNNING	NFETOVERCURRENTLIMIT	PFETOVERCURRENTLIMIT			VMONBUVDDRISE	VMONBUVDDFALL		R5VREADY	VMONIO1RISE	VMONIO1FALL	VMONIO0RISE	VMONIO0FALL	VMONDVDDRISE	VMONDVDDFALL	VMONALTAVDDRISE	VMONALTAVDDFALL	VMONAVDDRISE	VMONAVDDFALL

Bit	Name	Reset	Access	Description
31	TEMPHIGH	0	R	**Temperature High Limit Reached**
	Set when the value of a periodic temperature measurement is higher or equal than TEMPHIGH in EMU_TEMPLIMITS			
30	TEMPLOW	0	R	**Temperature Low Limit Reached**
	Set when the value of a periodic temperature measurement is lower or equal than TEMPHIGH in EMU_TEMPLIMITS			
29	TEMP	0	R	**New Temperature Measurement Valid**
	Set when a new periodic temperature measurement is available			
28:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	VSCALEDONE	0	R	**Voltage Scale Steps Done IRQ**
	Will be set when all the steps needed for voltage scaling is done. For voltage upgrade, the software can start increasing clock frequency after this interrupt. For voltage downgrade, this will indicate that hardware has finished all the steps including updating of BOD levels and regultor controls, but voltage drop may take lot longer due to load current			
24	EM23WAKEUP	0	R	**Wakeup IRQ From EM2 and EM3**
	Will be set when the system wakes up from EM2 and EM3. This interrupt can be used to run initialization code need to reconfigure the system when returning from EM2 and EM3.			
23	R5VVSINT	0	R	**5V Regulator Voltage Update Done**
	5V regulator voltage update done.			
22	BURDY	0	R	**Backup Functionality Ready Interrupt Flag**
	Set when the Backup functionality is ready for use.			
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	DCDCINBYPASS	0	R	**DCDC is in Bypass**
	DCDC is in bypass			
19	DCDCLNRUNNING	0	R	**LN Mode is Running**
	This flag is set once the DCDC regulator has started to run in LN mode			
18	DCDCLPRUNNING	0	R	**LP Mode is Running**
	This flag is set once the DCDC regulator has started to run in LP mode			

Bit	Name	Reset	Access	Description
17	NFETOVERCUR-RENTLIMIT	0	R	**NFET Current Limit Hit**
	Reserved for internal use.			
16	PFETOVERCUR-RENTLIMIT	0	R	**PFET Current Limit Hit**
	Reserved for internal use.			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	VMONBUVDDRISE	0	R	**VMON BUVDD Channel Rise**
	A rising edge on VMON BUVDD channel has been detected.			
12	VMONBUVDDFALL	0	R	**VMON BACKUP Channel Fall**
	A falling edge on VMON BUVDD channel has been detected.			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	R5VREADY	0	R	**5V Regulator is Ready to Use**
	5V regulator is ready to use.			
9	VMONIO1RISE	0	R	**VMON IOVDD1 Channel Rise**
	A rising edge on VMON IOVDD1 channel has been detected.			
8	VMONIO1FALL	0	R	**VMON IOVDD1 Channel Fall**
	A falling edge on VMON IOVDD1 channel has been detected.			
7	VMONIO0RISE	0	R	**VMON IOVDD0 Channel Rise**
	A rising edge on VMON IOVDD0 channel has been detected.			
6	VMONIO0FALL	0	R	**VMON IOVDD0 Channel Fall**
	A falling edge on VMON IOVDD0 channel has been detected.			
5	VMONDVDDRISE	0	R	**VMON DVDD Channel Rise**
	A rising edge on VMON DVDD channel has been detected.			
4	VMONDVDDFALL	0	R	**VMON DVDD Channel Fall**
	A falling edge on VMON DVDD channel has been detected.			
3	VMONALTAVDD-RISE	0	R	**Alternate VMON AVDD Channel Rise**
	A rising edge on Alternate VMON AVDD channel has been detected.			
2	VMONALTAVDD-FALL	0	R	**Alternate VMON AVDD Channel Fall**
	A falling edge on Alternate VMON AVDD channel has been detected.			
1	VMONAVDDRISE	0	R	**VMON AVDD Channel Rise**
	A rising edge on VMON AVDD channel has been detected.			
0	VMONAVDDFALL	0	R	**VMON AVDD Channel Fall**
	A falling edge on VMON AVDD channel has been detected.			

9.5.10 EMU_IFS - Interrupt Flag Set 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	0	0	0				0	0	0	0		0	0	0	0	0			0	0		0	0	0	0	0	0	0	0	0	0	0
Access	W1	W1	W1				W1	W1	W1	W1		W1	W1	W1	W1	W1			W1	W1		W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name	TEMPHIGH	TEMPLOW	TEMP				VSCALEDONE	EM23WAKEUP	R5VVSINT	BURDY		DCDCINBYPASS	DCDCLNRUNNING	DCDCLPRUNNING	NFETOVERCURRENTLIMIT	PFETOVERCURRENTLIMIT			VMONBUVDDRISE	VMONBUVDDFALL		R5VREADY	VMONIO1RISE	VMONIO1FALL	VMONIO0RISE	VMONIO0FALL	VMONDVDDRISE	VMONDVDDFALL	VMONALTAVDDRISE	VMONALTAVDDFALL	VMONAVDDRISE	VMONAVDDFALL

Bit	Name	Reset	Access	Description
31	TEMPHIGH	0	W1	**Set TEMPHIGH Interrupt Flag**
	Write 1 to set the TEMPHIGH interrupt flag			
30	TEMPLOW	0	W1	**Set TEMPLOW Interrupt Flag**
	Write 1 to set the TEMPLOW interrupt flag			
29	TEMP	0	W1	**Set TEMP Interrupt Flag**
	Write 1 to set the TEMP interrupt flag			
28:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	VSCALEDONE	0	W1	**Set VSCALEDONE Interrupt Flag**
	Write 1 to set the VSCALEDONE interrupt flag			
24	EM23WAKEUP	0	W1	**Set EM23WAKEUP Interrupt Flag**
	Write 1 to set the EM23WAKEUP interrupt flag			
23	R5VVSINT	0	W1	**Set R5VVSINT Interrupt Flag**
	Write 1 to set the R5VVSINT interrupt flag			
22	BURDY	0	W1	**Set BURDY Interrupt Flag**
	Write 1 to set the BURDY interrupt flag			
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	DCDCINBYPASS	0	W1	**Set DCDCINBYPASS Interrupt Flag**
	Write 1 to set the DCDCINBYPASS interrupt flag			
19	DCDCLNRUNNING	0	W1	**Set DCDCLNRUNNING Interrupt Flag**
	Write 1 to set the DCDCLNRUNNING interrupt flag			
18	DCDCLPRUNNING	0	W1	**Set DCDCLPRUNNING Interrupt Flag**
	Write 1 to set the DCDCLPRUNNING interrupt flag			

Bit	Name	Reset	Access	Description
17	NFETOVERCUR-RENTLIMIT	0	W1	**Set NFETOVERCURRENTLIMIT Interrupt Flag**
	Write 1 to set the NFETOVERCURRENTLIMIT interrupt flag			
16	PFETOVERCUR-RENTLIMIT	0	W1	**Set PFETOVERCURRENTLIMIT Interrupt Flag**
	Write 1 to set the PFETOVERCURRENTLIMIT interrupt flag			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	VMONBUVDDRISE	0	W1	**Set VMONBUVDDRISE Interrupt Flag**
	Write 1 to set the VMONBUVDDRISE interrupt flag			
12	VMONBUVDDFALL	0	W1	**Set VMONBUVDDFALL Interrupt Flag**
	Write 1 to set the VMONBUVDDFALL interrupt flag			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	R5VREADY	0	W1	**Set R5VREADY Interrupt Flag**
	Write 1 to set the R5VREADY interrupt flag			
9	VMONIO1RISE	0	W1	**Set VMONIO1RISE Interrupt Flag**
	Write 1 to set the VMONIO1RISE interrupt flag			
8	VMONIO1FALL	0	W1	**Set VMONIO1FALL Interrupt Flag**
	Write 1 to set the VMONIO1FALL interrupt flag			
7	VMONIO0RISE	0	W1	**Set VMONIO0RISE Interrupt Flag**
	Write 1 to set the VMONIO0RISE interrupt flag			
6	VMONIO0FALL	0	W1	**Set VMONIO0FALL Interrupt Flag**
	Write 1 to set the VMONIO0FALL interrupt flag			
5	VMONDVDDRISE	0	W1	**Set VMONDVDDRISE Interrupt Flag**
	Write 1 to set the VMONDVDDRISE interrupt flag			
4	VMONDVDDFALL	0	W1	**Set VMONDVDDFALL Interrupt Flag**
	Write 1 to set the VMONDVDDFALL interrupt flag			
3	VMONALTAVDD-RISE	0	W1	**Set VMONALTAVDDRISE Interrupt Flag**
	Write 1 to set the VMONALTAVDDRISE interrupt flag			
2	VMONALTAVDD-FALL	0	W1	**Set VMONALTAVDDFALL Interrupt Flag**
	Write 1 to set the VMONALTAVDDFALL interrupt flag			
1	VMONAVDDRISE	0	W1	**Set VMONAVDDRISE Interrupt Flag**
	Write 1 to set the VMONAVDDRISE interrupt flag			
0	VMONAVDDFALL	0	W1	**Set VMONAVDDFALL Interrupt Flag**
	Write 1 to set the VMONAVDDFALL interrupt flag			

9.5.11 EMU_IFC - Interrupt Flag Clear 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	0	0	0				0	0	0	0		0	0	0	0	0			0	0		0	0	0	0	0	0	0	0	0	0	0
Access	(R)W1	(R)W1	(R)W1				(R)W1	(R)W1	(R)W1	(R)W1		(R)W1	(R)W1	(R)W1	(R)W1	(R)W1			(R)W1	(R)W1		(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name	TEMPHIGH	TEMPLOW	TEMP				VSCALEDONE	EM23WAKEUP	R5VVSINT	BURDY		DCDCINBYPASS	DCDCLNRUNNING	DCDCLPRUNNING	NFETOVERCURRENTLIMIT	PFETOVERCURRENTLIMIT			VMONBUVDDRISE	VMONBUVDDFALL		R5VREADY	VMONIO1RISE	VMONIO1FALL	VMONIO0RISE	VMONIO0FALL	VMONDVDDRISE	VMONDVDDFALL	VMONALTAVDDRISE	VMONALTAVDDFALL	VMONAVDDRISE	VMONAVDDFALL

Bit	Name	Reset	Access	Description
31	TEMPHIGH	0	(R)W1	**Clear TEMPHIGH Interrupt Flag**
	Write 1 to clear the TEMPHIGH interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
30	TEMPLOW	0	(R)W1	**Clear TEMPLOW Interrupt Flag**
	Write 1 to clear the TEMPLOW interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
29	TEMP	0	(R)W1	**Clear TEMP Interrupt Flag**
	Write 1 to clear the TEMP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
28:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	VSCALEDONE	0	(R)W1	**Clear VSCALEDONE Interrupt Flag**
	Write 1 to clear the VSCALEDONE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
24	EM23WAKEUP	0	(R)W1	**Clear EM23WAKEUP Interrupt Flag**
	Write 1 to clear the EM23WAKEUP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
23	R5VVSINT	0	(R)W1	**Clear R5VVSINT Interrupt Flag**
	Write 1 to clear the R5VVSINT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
22	BURDY	0	(R)W1	**Clear BURDY Interrupt Flag**
	Write 1 to clear the BURDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	DCDCINBYPASS	0	(R)W1	**Clear DCDCINBYPASS Interrupt Flag**
	Write 1 to clear the DCDCINBYPASS interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
19	DCDCLNRUNNING	0	(R)W1	**Clear DCDCLNRUNNING Interrupt Flag**
	Write 1 to clear the DCDCLNRUNNING interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
18	DCDCLPRUNNING	0	(R)W1	**Clear DCDCLPRUNNING Interrupt Flag**
	Write 1 to clear the DCDCLPRUNNING interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
17	NFETOVERCUR-RENTLIMIT	0	(R)W1	**Clear NFETOVERCURRENTLIMIT Interrupt Flag**
	Write 1 to clear the NFETOVERCURRENTLIMIT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
16	PFETOVERCUR-RENTLIMIT	0	(R)W1	**Clear PFETOVERCURRENTLIMIT Interrupt Flag**
	Write 1 to clear the PFETOVERCURRENTLIMIT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	VMONBUVDDRISE	0	(R)W1	**Clear VMONBUVDDRISE Interrupt Flag**
	Write 1 to clear the VMONBUVDDRISE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
12	VMONBUVDDFALL	0	(R)W1	**Clear VMONBUVDDFALL Interrupt Flag**
	Write 1 to clear the VMONBUVDDFALL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	R5VREADY	0	(R)W1	**Clear R5VREADY Interrupt Flag**
	Write 1 to clear the R5VREADY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	VMONIO1RISE	0	(R)W1	**Clear VMONIO1RISE Interrupt Flag**
	Write 1 to clear the VMONIO1RISE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	VMONIO1FALL	0	(R)W1	**Clear VMONIO1FALL Interrupt Flag**
	Write 1 to clear the VMONIO1FALL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	VMONIO0RISE	0	(R)W1	**Clear VMONIO0RISE Interrupt Flag**
	Write 1 to clear the VMONIO0RISE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
6	VMONIO0FALL	0	(R)W1	**Clear VMONIO0FALL Interrupt Flag**
	Write 1 to clear the VMONIO0FALL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
5	VMONDVDDRISE	0	(R)W1	**Clear VMONDVDDRISE Interrupt Flag**
	Write 1 to clear the VMONDVDDRISE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
4	VMONDVDDFALL	0	(R)W1	**Clear VMONDVDDFALL Interrupt Flag**
	Write 1 to clear the VMONDVDDFALL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	VMONALTAVDD-RISE	0	(R)W1	**Clear VMONALTAVDDRISE Interrupt Flag**
	Write 1 to clear the VMONALTAVDDRISE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	VMONALTAVDD-FALL	0	(R)W1	**Clear VMONALTAVDDFALL Interrupt Flag**
	Write 1 to clear the VMONALTAVDDFALL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	VMONAVDDRISE	0	(R)W1	**Clear VMONAVDDRISE Interrupt Flag**
	Write 1 to clear the VMONAVDDRISE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	VMONAVDDFALL	0	(R)W1	**Clear VMONAVDDFALL Interrupt Flag**
	Write 1 to clear the VMONAVDDFALL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

9.5.12 EMU_IEN - Interrupt Enable 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	0	0	0				0	0	0	0		0	0	0	0	0			0	0		0	0	0	0	0	0	0	0	0	0	0
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
Name	TEMPHIGH	TEMPLOW	TEMP				VSCALEDONE	EM23WAKEUP	R5VVSINT	BURDY		DCDCINBYPASS	DCDCLNRUNNING	DCDCLPRUNNING	NFETOVERCURRENTLIMIT	PFETOVERCURRENTLIMIT			VMONBUVDDRISE	VMONBUVDDFALL		R5VREADY	VMONIO1RISE	VMONIO1FALL	VMONIO0RISE	VMONIO0FALL	VMONDVDDRISE	VMONDVDDFALL	VMONALTAVDDRISE	VMONALTAVDDFALL	VMONAVDDRISE	VMONAVDDFALL

Bit	Name	Reset	Access	Description
31	TEMPHIGH	0	RW	**TEMPHIGH Interrupt Enable**
	Enable/disable the TEMPHIGH interrupt			
30	TEMPLOW	0	RW	**TEMPLOW Interrupt Enable**
	Enable/disable the TEMPLOW interrupt			
29	TEMP	0	RW	**TEMP Interrupt Enable**
	Enable/disable the TEMP interrupt			
28:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	VSCALEDONE	0	RW	**VSCALEDONE Interrupt Enable**
	Enable/disable the VSCALEDONE interrupt			
24	EM23WAKEUP	0	RW	**EM23WAKEUP Interrupt Enable**
	Enable/disable the EM23WAKEUP interrupt			
23	R5VVSINT	0	RW	**R5VVSINT Interrupt Enable**
	Enable/disable the R5VVSINT interrupt			
22	BURDY	0	RW	**BURDY Interrupt Enable**
	Enable/disable the BURDY interrupt			
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	DCDCINBYPASS	0	RW	**DCDCINBYPASS Interrupt Enable**
	Enable/disable the DCDCINBYPASS interrupt			
19	DCDCLNRUNNING	0	RW	**DCDCLNRUNNING Interrupt Enable**
	Enable/disable the DCDCLNRUNNING interrupt			
18	DCDCLPRUNNING	0	RW	**DCDCLPRUNNING Interrupt Enable**
	Enable/disable the DCDCLPRUNNING interrupt			

Bit	Name	Reset	Access	Description
17	NFETOVERCUR-RENTLIMIT	0	RW	**NFETOVERCURRENTLIMIT Interrupt Enable**
	Enable/disable the NFETOVERCURRENTLIMIT interrupt			
16	PFETOVERCUR-RENTLIMIT	0	RW	**PFETOVERCURRENTLIMIT Interrupt Enable**
	Enable/disable the PFETOVERCURRENTLIMIT interrupt			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	VMONBUVDDRISE	0	RW	**VMONBUVDDRISE Interrupt Enable**
	Enable/disable the VMONBUVDDRISE interrupt			
12	VMONBUVDDFALL	0	RW	**VMONBUVDDFALL Interrupt Enable**
	Enable/disable the VMONBUVDDFALL interrupt			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	R5VREADY	0	RW	**R5VREADY Interrupt Enable**
	Enable/disable the R5VREADY interrupt			
9	VMONIO1RISE	0	RW	**VMONIO1RISE Interrupt Enable**
	Enable/disable the VMONIO1RISE interrupt			
8	VMONIO1FALL	0	RW	**VMONIO1FALL Interrupt Enable**
	Enable/disable the VMONIO1FALL interrupt			
7	VMONIO0RISE	0	RW	**VMONIO0RISE Interrupt Enable**
	Enable/disable the VMONIO0RISE interrupt			
6	VMONIO0FALL	0	RW	**VMONIO0FALL Interrupt Enable**
	Enable/disable the VMONIO0FALL interrupt			
5	VMONDVDDRISE	0	RW	**VMONDVDDRISE Interrupt Enable**
	Enable/disable the VMONDVDDRISE interrupt			
4	VMONDVDDFALL	0	RW	**VMONDVDDFALL Interrupt Enable**
	Enable/disable the VMONDVDDFALL interrupt			
3	VMONALTAVDD-RISE	0	RW	**VMONALTAVDDRISE Interrupt Enable**
	Enable/disable the VMONALTAVDDRISE interrupt			
2	VMONALTAVDD-FALL	0	RW	**VMONALTAVDDFALL Interrupt Enable**
	Enable/disable the VMONALTAVDDFALL interrupt			
1	VMONAVDDRISE	0	RW	**VMONAVDDRISE Interrupt Enable**
	Enable/disable the VMONAVDDRISE interrupt			
0	VMONAVDDFALL	0	RW	**VMONAVDDFALL Interrupt Enable**
	Enable/disable the VMONAVDDFALL interrupt			

9.5.13 EMU_PWRLOCK - Regulator and Supply Lock 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																	0x0000															
Access																	RW															
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	0x0000	RW	**Regulator and Supply Configuration Lock Key**

Write any other value than the unlock code to lock all regulator control registers, from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled. Registers that are locked: PWRCFG, PWRCTRL and DCDC* registers.

Mode	Value	Description
Read Operation		
UNLOCKED	0	EMU Regulator registers are unlocked
LOCKED	1	EMU Regulator registers are locked
Write Operation		
LOCK	0	Lock EMU Regulator registers
UNLOCK	0xADE8	Unlock EMU Regulator registers

9.5.14 EMU_PWRCTRL - Power 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																			0			0					0					
Access																			RW			RW					RW					
Name																			IMMEDIATEPWRSWITCH			REGPWRSEL					ANASW					

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	IMMEDIATEPWRSWITCH	0	RW	**Allows Immediate Switching of ANASW and REGPWRSEL Bitfields** When set, allows immediate ANASW/REGPWRSEL switching. When cleared, Hardware protects switching of ANASW/REGPWRSEL and switching is applied by hardware only when DCDC is stable
12:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	REGPWRSEL	0	RW	**This Field Selects the Input Supply Pin for the Digital LDO** Determines the power supply input used by the Digital LDO. Firmware should select DVDD as the input after startup. If DCDC is not configured to drive DVDD, IMMEDIATEPWRSWITCH needs to be set to prior to setting this bit to immediately make the switch. If DCDC is configured to drive DVDD, hardware will make the switch to DVDD only when DCDC is stable

Value	Mode	Description
0	AVDD	The AVDD pin is the supply for the digital LDO. LDO current is limited to 20 mA in this configuration.
1	DVDD	The DVDD pin is the supply for the digital LDO. Firmware should set REGPWRSEL=1 after startup, before increasing the core clock frequency.

9:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	ANASW	0	RW	**Analog Switch Selection** Determines the power supply routed to the analog supply (VDDX_ANA) used by the analog peripherals (e.g., ULFRCO, LFRCO, LFXO, HFRCO, AUXHFRCO, VMON, IDAC, and ADC). Reset with POR, Hard Pin Reset, or BOD Reset. If DCDC is not configured to drive DVDD, IMMEDIATEPWRSWITCH needs to be set to prior to setting this bit to immediately make the switch. If DCDC is configured to drive DVDD, hardware will make the switch to DVDD only when DCDC is stable

Value	Mode	Description
0	AVDD	Select AVDD as the analog power supply
1	DVDD	Select DVDD as the analog power supply

4:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

9.5.15 EMU_DCDCCTRL - DCDC 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																											1	1			0x3	
Access																											RW	RW			RW	
Name																											DCDCMODEEM4	DCDCMODEEM23			DCDCMODE	

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	DCDCMODEEM4	1	RW	**DCDC Mode EM4H**

Determines the DCDC mode in EM4H. This bit is ignored if DCDCMODE=Bypass. Reset with POR, Hard Pin Reset, or BOD Reset.

Value	Mode	Description
0	EM4SW	DCDC mode is according to DCDCMODE field.
1	EM4LOWPOWER	DCDC mode is low power.

Bit	Name	Reset	Access	Description
4	DCDCMODEEM23	1	RW	**DCDC Mode EM23**

Determines the DCDC mode in EM2 and EM3. This bit is ignored if DCDCMODE=Bypass. Reset with POR, Hard Pin Reset, or BOD Reset.

Value	Mode	Description
0	EM23SW	DCDC mode is according to DCDCMODE field.
1	EM23LOWPOWER	DCDC mode is low power.

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	DCDCMODE	0x3	RW	**Regulator Mode**

Determines the operating mode of the DCDC regulator. Reset with POR, Hard Pin Reset, or BOD Reset.

Value	Mode	Description
0	BYPASS	DCDC regulator is operating in bypass mode. Prior to configuring DCDCMODE=BYPASS, software must set EMU_DCDCCLIMCTRL.BYPLIMEN=1 to prevent excessive current between VREGVDD and DVDD supplies.
1	LOWNOISE	DCDC regulator is operating in low noise mode.
2	LOWPOWER	DCDC regulator is operating in low power mode.
3	OFF	DCDC regulator is off and the bypass switch is off. Note: DVDD must be supplied externally

9.5.16 EMU_DCDCMISCCTRL - DCDC Miscellaneous Control 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			0x3				0x1			0x0				0x7				0x7						0			1	1	0
Access			RW			RW				RW			RW				RW				RW						RW			RW	RW	RW
Name			LPCMPBIASEM234H			LNCLIMILIMSEL				LPCLIMILIMSEL			BYPLIMSEL				NFETCNT				PFETCNT						LNFORCECCMIMM			LPCMPHYSHI	LPCMPHYSDIS	LNFORCECCM

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	LPCMPBIASEM234H	0x0	RW	**LP Mode Comparator Bias Selection for EM23 or EM4H**

LP mode comparator bias selection. Reset with POR, Hard Pin Reset, or BOD Reset.

Value	Mode	Description
0	BIAS0	Maximum load current less than 75uA.
1	BIAS1	Maximum load current less than 500uA.
2	BIAS2	Maximum load current less than 2.5mA.
3	BIAS3	Maximum load current less than 10mA.

27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:24	LNCLIMILIMSEL	0x3	RW	**Current Limit Level Selection for Current Limiter in LN Mode**

High-side current limiter's current limit level selection in low noise mode. The recommended setting is calculated by LNCLIMILIMSEL=(I_MAX+40mA)*1.5/(5mA*(PFETCNT+1))-1, where I_MAX is the maximum average current allowed to the load, and 40mA represents the current ripple with some margin, and the factor of 1.5 accounts for detecting error and other variations. For strong (i.e., low internal impedance) battery, it is recommended to have I_MAX=200mA. I_MAX should never be set higher than 200mA to avoid reliability issues. Reset with POR, Hard Pin Reset, or BOD reset.

23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	LPCLIMILIMSEL	0x1	RW	**Current Limit Level Selection for Current Limiter in LP Mode**

Sets high-side current limit in low power mode, with maxixum current equal to 40 mA*(1+LPCLIMILIMSEL). Recommend setting LPCLIMILIMSEL=1, corresponding to a maximum current of 80 mA for optimal efficiency and to support up to 10 mA loads when LPCMPBIASEM234H=0x3. Reset with POR, Hard Pin Reset, or BOD reset.

19:16	BYPLIMSEL	0x0	RW	**Current Limit in Bypass Mode**

Set current limit in bypass mode when BYPLIMEN equals one. The limit is from 20mA to 320mA, with 20mA/step. Reset with POR, Hard Pin Reset, or BOD Reset.

15:12	NFETCNT	0x7	RW	**NFET Switch Number Selection**

Low Noise mode NFET power switch count number. The selected number of switches are NFETCNT+1. This may cause a very momentary efficiency hit. Reset with POR, Hard Pin Reset, or BOD Reset.

Bit	Name	Reset	Access	Description
11:8	PFETCNT	0x7	RW	**PFET Switch Number Selection**
	Low Noise mode PFET power switch count number. The selected number of switches are PFETCNT+1. Reset with POR, Hard Pin Reset, or BOD Reset.			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	LNFORCECCMIMM	0	RW	**Force DCDC Into CCM Mode Immediately, Based on LNFORCECCM**
	When set, this bit allows software to change LNFORCECCM bit and have the change take effect while DCDC is running. Otherwise, LNFORCECCM must be programmed prior to enabling the DCDC.			
4:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	LPCMPHYSHI	1	RW	**Comparator Threshold on the High Side**
	Reserved for internal use. Should always be set to 1.			
1	LPCMPHYSDIS	1	RW	**Disable LP Mode Hysteresis in the State Machine Control**
	Reserved for internal use. Should always be set to 1.			
0	LNFORCECCM	0	RW	**Force DCDC Into CCM Mode in Low Noise Operation**
	When this bit is set to 0 in low noise mode, the zero detector is configured as zero-crossing detector and the DCDC will be in forced CCM mode. The threshold set by ZDETILIMSEL will be ignored. When this bit is set to 1 in low noise mode, the zero detector is configured as reverse-current limiter and the DCDC will be in DCM mode. The reverse current limit level is set by ZDETILIMSEL. In low power mode, the zero detector is always configured as zero-crossing detector. Reset with POR, Hard Pin Reset, or BOD reset.			

9.5.17 EMU_DCDCZDETCTRL - DCDC Power Train NFET Zero Current Detector 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																							0x1			0x5						
Access																							RW			RW						
Name																							ZDETBLANKDLY			ZDETILIMSEL						

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	ZDETBLANKDLY	0x1	RW	**Reserved for internal use. Do not change.** Reserved for internal use. Do not change.
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	ZDETILIMSEL	0x5	RW	**Reverse Current Limit Level Selection for Zero Detector** Zero detector is reconfigured as low-side reverse current limiter when LNFORCECCM=1 in LN mode. The configuration of this register is calculated by the allowed average reverse current I_RMAX through the equation: ZDETILIMSEL=(I_RMAX +40mA)*1.5/(2.5mA*(NFETCNT+1)), where 40mA represents the current ripple with some margin, and the factor of 1.5 accounts for detecting error and other variations. When the battery can tolerate large reverse current, it is recommended to have I_RMAX=160mA to maximize ZDETILIMSEL to 7 with NFETCNT=15. Note that when LNFORCECCM=1 but ZDETILIMSEL=0, the DCDC's behavior will be very similar to when LNFORCECCM=0 - that is, the DCDC will be in DCM mode. When LNFORCECCM=0, the zero detector will only detect zero-crossings (reverse-current limit=0 mA) and this register is ignored. Reset with POR, Hard Pin Reset, or BOD reset.
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

9.5.18 EMU_DCDCCLIMCTRL - DCDC Power Train PFET Current Limiter 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																			0				0x1									
Access																			RW				RW									
Name																			BYPLIMEN				CLIMBLANKDLY									

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	BYPLIMEN	0	RW	**Bypass Current Limit Enable**
	Bypass current limit enable. Setting this bit limits maximum current drawn from DCDC input supply while DCDC is in BYPASS mode. Note that the device will see an additional ~10 µA of current draw when BYPLIMEN=1 and Bypass Mode is enabled. To prevent this excess current, applications should disable the Bypass Current Limit (BYPLIMEN=0) once the DVDD voltage has reached the main supply voltage in Bypass Mode. Reset with POR, Hard Pin Reset, or BOD Reset.			
12:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	CLIMBLANKDLY	0x1	RW	**Reserved for internal use. Do not change.**
	Reserved for internal use. Do not change.			
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

9.5.19 EMU_DCDCLNCOMPCTRL - DCDC Low Noise Compensator Control 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	0x5					0x7					0x2						0x4							0x07						0x7		
Access	RW					RW					RW						RW							RW						RW		
Name	COMPENC3					COMPENC2					COMPENC1						COMPENR3							COMPENR2						COMPENR1		

Bit	Name	Reset	Access	Description
31:28	COMPENC3	0x5	RW	**Low Noise Mode Compensator C3 Trim Value**
	LN mode compensator C3 trim, 0.5pF-8pF in 0.5pF steps. Reset with POR, Hard Pin Reset, or BOD Reset.			
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:24	COMPENC2	0x7	RW	**Low Noise Mode Compensator C2 Trim Value**
	LN mode compensator C2 trim, 1pF-8pF in 1pF steps. Reset with POR, Hard Pin Reset, or BOD Reset.			
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	COMPENC1	0x2	RW	**Low Noise Mode Compensator C1 Trim Value**
	LN mode compensator C1 trim, 0.15pF-0.60pF in 0.15pF step. Reset with POR, Hard Pin Reset, or BOD Reset.			
19:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:12	COMPENR3	0x4	RW	**Low Noise Mode Compensator R3 Trim Value**
	LN mode compensator r3 trim, 5-80KOhm in 5Khom steps. Reset with POR, Hard Pin Reset, or BOD Reset.			
11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:4	COMPENR2	0x07	RW	**Low Noise Mode Compensator R2 Trim Value**
	LN mode compensator r2 trim, 50-1600KOhm, in 50KOhm steps. Reset with POR, Hard Pin Reset, or BOD Reset.			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	COMPENR1	0x7	RW	**Low Noise Mode Compensator R1 Trim Value**
	LN mode compensator r1 trim, 500-1200kOhm, in 100KOhm steps. Reset with POR, Hard Pin Reset, or BOD Reset.			

9.5.20 EMU_DCDCLNVCTRL - DCDC Low Noise Voltage 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																					0x71										0	
Access																					RWH										RW	
Name																					LNVREF										LNATT	

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	LNVREF	0x71	RWH	**Low Noise Mode VREF Trim**
	Low noise mode Vref trim. LNATT and LNVREF set the output of the DCDC to 3*(1+LNATT)*(235.48+3.226*LNVREF). Customers should use the emlib functions for configuring this field. Reset with POR, Hard Pin Reset, or BOD Reset.			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	LNATT	0	RW	**Low Noise Mode Feedback Attenuation**
	Low noise mode feedback attenuation. Customers should use the emlib functions for configuring this field. Reset with POR, Hard Pin Reset, or BOD Reset.			

Value	Mode	Description
0	DIV3	Feedback Ratio is 1/3
1	DIV6	Feedback Ratio is 1/6

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

9.5.21 EMU_DCDCLPVCTRL - DCDC Low Power Voltage 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																											0xB4					0
Access																											RW					RW
Name																											LPVREF					LPATT

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*		
8:1	LPVREF	0xB4	RW	**LP Mode Reference Selection for EM23 and EM4H**
	Select Vref level. Maximum available code is 8'b11100111. LPATT and LPVREFSEL set the output of the DCDC to 4*(1+LPATT)*(30+LPVREF)*2.2mV. Customers should use the emlib functions for configuring this field. Reset with POR, Hard Pin Reset, or BOD Reset.			
0	LPATT	0	RW	**Low Power Feedback Attenuation**
	Low power feedback attenuation select. Customers should use the emlib functions for configuring this field. Reset with POR, Hard Pin Reset, or BOD Reset.			

Value	Mode	Description
0	DIV4	Feedback Ratio is 1/4
1	DIV8	Feedback Ratio is 1/8

9.5.22 EMU_DCDCLPCTRL - DCDC Low Power Control Register

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						0x1		1									0x0															
Access						RW		RW									RW															
Name						LPBLANK		LPVREFDUTYEN									LPCMPHYSSELEM234H															

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*		
26:25	LPBLANK	0x1	RW	**Reserved for internal use. Do not change.**
	Reserved for internal use. Do not change.			
24	LPVREFDUTYEN	1	RW	**LP Mode Duty Cycling Enable**
	Allow duty cycling of the bias. This is to minimize DC bias. Reset with POR, Hard Pin Reset, or BOD Reset.			
23:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:12	LPCMPHYSSELEM234H	0x0	RW	**LP Mode Hysteresis Selection for EM23 and EM4H**
	User-programmable hysteresis level for the low power comparator. Hysteresis voltage at the output is 4*(1+LPATT)*LPCMPHYSSEL*3.13mv. Customers should use the emlib functions for configuring this field. Reset with POR, Hard Pin Reset, or BOD Reset.			
11:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

9.5.23 EMU_DCDCLNFREQCTRL - DCDC Low Noise Controller Frequency 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				0x10																											0x0	
Access				RW																											RW	
Name				RCOTRIM																											RCOBAND	

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	RCOTRIM	0x10	RW	**Reserved for internal use. Do not change.**
	Reserved for internal use. Do not change.			
23:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	RCOBAND	0x0	RW	**LN Mode RCO Frequency Band Selection**
	Low noise mode RCO frequency selection. 0~7: 3~8.95MHz, approximately 0.85MHz/step. Reset with POR, Hard Pin Reset, or BOD Reset.			

9.5.24 EMU_DCDCSYNC - DCDC Read Status 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																																0
Access																																R
Name																																DCDCCTRLBUSY

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	DCDCCTRLBUSY	0	R	**DCDC CTRL Register Transfer Busy**
	Indicates the status of the DCDCCTRL transfer to the EMU OSC clock domain. Software cannot re-write the DCDCCTRL register until this signal goes low.			

9.5.25 EMU_VMONAVDDCTRL - VMON AVDD Channel 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								0	0		0
Access									RW				RW				RW				RW								RW	RW		RW
Name									RISETHRESCOARSE				RISETHRESFINE				FALLTHRESCOARSE				FALLTHRESFINE								FALLWU	RISEWU		EN

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*		
23:20	RISETHRESCOARSE	0x0	RW	**Rising Threshold Coarse Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
19:16	RISETHRESFINE	0x0	RW	**Rising Threshold Fine Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
15:12	FALLTHRESCOARSE	0x0	RW	**Falling Threshold Coarse Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
11:8	FALLTHRESFINE	0x0	RW	**Falling Threshold Fine Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	FALLWU	0	RW	**Fall Wakeup**
	When set, a wakeup from EM4H will take place upon a falling edge. Reset with SYSEXTENDEDRESETn.			
2	RISEWU	0	RW	**Rise Wakeup**
	When set, a wakeup from EM4H will take place upon a rising edge. Reset with SYSEXTENDEDRESETn.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**Enable**
	Set this bit to enable the AVDD VMON. Reset with SYSEXTENDEDRESETn.			

9.5.26 EMU_VMONALTAVDDCTRL - Alternate VMON AVDD Channel Control

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								0	0		0
Access																	RW				RW								RW	RW		RW
Name																	THRESCOARSE				THRESFINE								FALLWU	RISEWU		EN

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	THRESCOARSE	0x0	RW	**Threshold Coarse Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
11:8	THRESFINE	0x0	RW	**Threshold Fine Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	FALLWU	0	RW	**Fall Wakeup**
	When set, a wakeup from EM4H will take place upon a falling edge. Reset with SYSEXTENDEDRESETn.			
2	RISEWU	0	RW	**Rise Wakeup**
	When set, a wakeup from EM4H will take place upon a rising edge. Reset with SYSEXTENDEDRESETn.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**Enable**
	Set this bit to enable the ALTAVDD VMON. Reset with SYSEXTENDEDRESETn.			

9.5.27 EMU_VMONDVDDCTRL - VMON DVDD Channel Control

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								0	0		0
Access																	RW				RW								RW	RW		RW
Name																	THRESCOARSE				THRESFINE								FALLWU	RISEWU		EN

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	THRESCOARSE	0x0	RW	**Threshold Coarse Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
11:8	THRESFINE	0x0	RW	**Threshold Fine Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	FALLWU	0	RW	**Fall Wakeup**
	When set, a wakeup from EM4H will take place upon a falling edge. Reset with SYSEXTENDEDRESETn.			
2	RISEWU	0	RW	**Rise Wakeup**
	When set, a wakeup from EM4H will take place upon a rising edge. Reset with SYSEXTENDEDRESETn.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**Enable**
	Set this bit to enable the DVDD VMON. Reset with SYSEXTENDEDRESETn.			

9.5.28 EMU_VMONIO0CTRL - VMON IOVDD0 Channel Control

Offset	Bit Position																													
0x09C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0				0x0							0	0	0		0
Access																	RW				RW							RW	RW	RW		RW
Name																	THRESCOARSE				THRESFINE							RETDIS	FALLWU	RISEWU		EN

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	THRESCOARSE	0x0	RW	**Threshold Coarse Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
11:8	THRESFINE	0x0	RW	**Threshold Fine Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	RETDIS	0	RW	**EM4 IO0 Retention Disable**
	When set, the IO0 Retention will be disabled when this IO0 voltage drops below the threshold set. Reset with SYSEXTENDEDRESETn.			
3	FALLWU	0	RW	**Fall Wakeup**
	When set, a wakeup from EM4H will take place upon a falling edge. Reset with SYSEXTENDEDRESETn.			
2	RISEWU	0	RW	**Rise Wakeup**
	When set, a wakeup from EM4H will take place upon a rising edge. Reset with SYSEXTENDEDRESETn.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**Enable**
	Set this bit to enable the IO0 VMON. Reset with SYSEXTENDEDRESETn.			

9.5.29 EMU_VMONIO1CTRL - VMON IOVDD1 Channel Control

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							0	0	0		0
Access																	RW				RW							RW	RW	RW		RW
Name																	THRESCOARSE				THRESFINE							RETDIS	FALLWU	RISEWU		EN

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	THRESCOARSE	0x0	RW	**Threshold Coarse Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
11:8	THRESFINE	0x0	RW	**Threshold Fine Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	RETDIS	0	RW	**EM4 IO1 Retention Disable**
	When set, the IO1 Retention will be disabled when this IO1 voltage drops below the threshold set. Reset with SYSEXTENDEDRESETn.			
3	FALLWU	0	RW	**Fall Wakeup**
	When set, a wakeup from EM4H will take place upon a falling edge. Reset with SYSEXTENDEDRESETn.			
2	RISEWU	0	RW	**Rise Wakeup**
	When set, a wakeup from EM4H will take place upon a rising edge. Reset with SYSEXTENDEDRESETn.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**Enable**
	Set this bit to enable the IO1 VMON. Reset with SYSEXTENDEDRESETn.			

9.5.30 EMU_VMONBUVDDCTRL - VMON BUVDD Channel Control

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								0	0		0
Access																	RW				RW								RW	RW		RW
Name																	THRESCOARSE				THRESFINE								FALLWU	RISEWU		EN

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	THRESCOARSE	0x0	RW	**Threshold Coarse Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
11:8	THRESFINE	0x0	RW	**Threshold Fine Adjust**
	Check VMON section for programming the threshold value. Reset with SYSEXTENDEDRESETn.			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	FALLWU	0	RW	**Fall Wakeup**
	When set, a wakeup from EM4H will take place upon a falling edge. Reset with SYSEXTENDEDRESETn.			
2	RISEWU	0	RW	**Rise Wakeup**
	When set, a wakeup from EM4H will take place upon a rising edge. Reset with SYSEXTENDEDRESETn.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**Enable**
	Set this bit to enable the BUVDD VMON.			

9.5.31 EMU_RAM1CTRL - Memory Control Register

Offset	Bit Position																															
0x0B4	31	30	29	28	27	26	25	24	23	22	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																													RAMPOWERDOWN			

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	RAMPOWERDOWN	0x0	RW	**RAM1 Blockset Power-down**

RAM blockset power-down in EM23 with full access in EM01.

Value	Mode	Description
0	NONE	None of the RAM blocks powered down
8	BLK3	Power down RAM block 3 (address range 0x2001C000-0x2001FFFF)
12	BLK2TO3	Power down RAM blocks 2-3 (address range 0x20018000-0x2001FFFF)
14	BLK1TO3	Power down RAM blocks 1-3 (address range 0x20014000-0x2001FFFF)
15	BLK0TO3	Power down RAM blocks 0-3 (address range 0x20010000-0x2001FFFF)

9.5.32 EMU_RAM2CTRL - Memory Control Register

Offset	Bit Position																															
0x0B8	31	30	29	28	27	26	25	24	23	22	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																													RAMPOWERDOWN			

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	RAMPOWERDOWN	0x0	RW	**RAM2 Blockset Power-down**

RAM blockset power-down in EM23 with full access in EM01.

Mode	Value	Description
NONE	0x00	None of the RAM blocks powered down
BLK3	0x8	Power down RAM block 3 (address range 0x2002C000-0x2002FFFF)
BLK2TO3	0xC	Power down RAM blocks 2-3 (address range 0x20028000-0x2002FFFF)
BLK1TO3	0xE	Power down RAM blocks 1-3 (address range 0x20024000-0x2002FFFF)
BLK0TO3	0xF	Power down RAM blocks 0-3 (address range 0x20020000-0x2002FFFF)

9.5.33 EMU_BUCTRL - Backup Power Configuration Register

Offset	Bit Position																														
0x0BC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0										0x0				0x0				0x0				0x0							0	0	0
Access	RW										RW				RW				RW				RW							RW	RW	RW
Name	DISMAXCOMP										BUINACTPWRCON				BUACTPWRCON				PWRRES				VOUTRES							BUVINPROBEEN	STATEN	EN

Bit	Name	Reset	Access	Description
31	DISMAXCOMP	0	RW	**Disable MAIN-BU Comparator**
	Should be set to 1 if no backup battery is connected.			
30:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	BUINACTPWRCON	0x0	RW	**Power Connection Configuration When Not in Backup Mode**

Value	Mode	Description
0	NONE	No connection.
1	MAINBU	Main power and backup power are connected through a diode, allowing current to flow from main power source to backup power source, but not the other way.
2	BUMAIN	Main power and backup power are connected through a diode, allowing current to flow from backup power source to main power source, but not the other way.
3	NODIODE	Main power and backup power are connected without diode.

19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	BUACTPWRCON	0x0	RW	**Power Connection Configuration in Backup Mode**

Value	Mode	Description
0	NONE	No connection.
1	MAINBU	Main power and backup power are connected through a diode, allowing current to flow from backup power source to main power source, but not the other way.
2	BUMAIN	Main power and backup power are connected through a diode, allowing current to flow from main power source to backup power source, but not the other way.
3	NODIODE	Main power and backup power are connected without diode.

15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
13:12	PWRRES	0x0	RW	**Power Domain Resistor Select**

Select value of series resistor between main power domain and backup power domain.

Value	Mode	Description
0	RES0	Main power and backup power connected with RES0 series resistance.
1	RES1	Main power and backup power connected with RES1 series resistance.
2	RES2	Main power and backup power connected with RES2 series resistance.
3	RES3	Main power and backup power connected with RES3 series resistance.

11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	VOUTRES	0x0	RW	**BU_VOUT Resistor Select**

Disconnect or select resistance between backup domain power supply (AVDD in main mode and BU_VIN in backup mode) and BU_VOUT.

Value	Mode	Description
0	DIS	BU_VOUT is not connected
1	WEAK	Enable weak switch between BU_VOUT and backup domain power supply.
2	MED	Enable medium switch between BU_VOUT and backup domain power supply.
3	STRONG	Enable strong switch between BU_VOUT and backup domain power supply.

7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	BUVINPROBEEN	0	RW	**Enable BU_VIN Probing**

When enabled, BU_VIN/8 is generated (to be measured using the ADC).

1	STATEN	0	RW	**Enable Backup Mode Status Export**

When enabled, BU_STAT will indicate when backup mode is active.

0	EN	0	RW	**Enable Backup Mode**

Backup mode will be entered when main power browns out and backup battery is present.

9.5.34 EMU_R5VCTRL - 5V Regulator Control

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							0	0	0
Access																							RW							RW	RW	RW
Name																							INPUTMODE							IMONEN	EM4WUEN	BYPASS

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	INPUTMODE	0x0	RW	**5V Input Mode**
	Select input power supply of 5V regulator. Note that when the regulator BYPASS bit is enabled, firmare should not use the INPUTMODE = AUTO.			

Value	Mode	Description
0	AUTO	Regulator input supply switched automatically to the highest voltage of either VBUS or VREGI
1	VBUS	Force VBUS pin as the regulator input
2	VREGI	Force VREGI pin as the regulator input

7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	IMONEN	0	RW	**Enable the Regulator Current Monitor for Selected Current Path to Either VREGI or VBUS**
	When set enables the regulator current monitor for selected current path to either VREGI or VBUS.			
1	EM4WUEN	0	RW	**Enable EM4 Wakeup Due to VBUS Detection**
	Enable EM4 wakeup due to VBUS detection			
0	BYPASS	0	RW	**5V Regulator Bypass**
	Puts the 5V regulator in bypass mode, effectively shorting the regulator output to its input. This mode should only be enabled when the regulator input is less than or equal to 3.8V. Enabling BYPASS with > 3.8V may damage the chip. If BYPASS is enabled, INPUTMODE should be set to either VBUS or VREGI, and not AUTO.			

9.5.35 EMU_R5VADCCTRL - 5V Regulator Control

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															0
Access																	RW															RW
Name																	AMUXSEL															ENAMUX

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	AMUXSEL	0x0	RW	**ADC Mux Selection**
	ADC mux selection.			

Value	Mode	Description
0	VBUSDIV10	VBUS divided by 10
1	VREGIDIV10	VREGI divided by 10
2	VREGODIV6	VREGO divided by 6
3	VREGIIMON	VREGI current monitor
4	VBUSIMON	VBUS current monitor

11:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	ENAMUX	0	RW	**Enable the 5V Subsystem ADC MUX**
	When set, enables the 5V subsystem ADC MUX.			

9.5.36 EMU_R5VOUTLEVEL - 5V Regulator Voltage Select

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																													0x1		
Access																													RW		
Name																													OUTLEVEL		

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	OUTLEVEL	0x1	RW	**5V Regulator Voltage**
	Determines the output voltage of the 5V regulator. The equation for the regulator output is given by 2.3 + OUTLEVEL*0.1. The OUTLEVEL should not be set to zero as the regulator does not support it.			

9.5.37 EMU_R5VDETCTRL - 5V Detector Enables

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																														0	0	0
Access																														RW	RW	RW
Name																														VREGODETDIS	VBUSDETDIS	VREGIDETDIS

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	VREGODETDIS	0	RW	**VREGO Detector Disable**
	Disable VREGO Detector			
1	VBUSDETDIS	0	RW	**VBUS Detector Disable**
	Disable VBUS Detector			
0	VREGIDETDIS	0	RW	**VREGI Detector Disable**
	Disable VREGI Detector			

9.5.38 EMU_DCDCLPEM01CFG - Configuration Bits for Low Power Mode to Be Applied During EM01, This Field is Only Relevant If LP Mode is Used in EM01

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						0x3									
Access																	RW						RW									
Name																	LPCMPHYSSELEM01						LPCMPBIASEM01									

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	LPCMPHYSSELEM01	0x0	RW	**LP Mode Hysteresis Selection for EM01** LP mode hysteresis voltage in at the output is 4*(1+LPATT)*LPCMPHYSSEL*3.13mV. Customers should use the emlib functions for configuring this field.
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	LPCMPBIASEM01	0x3	RW	**LP Mode Comparator Bias Selection for EM01** Reserved for internal use. Do not change.

Value	Mode	Description
0	BIAS0	Maximum load current less than 75uA.
1	BIAS1	Maximum load current less than 500uA.
2	BIAS2	Maximum load current less than 2.5mA.
3	BIAS3	Maximum load current less than 10mA.

7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

9.5.39 EMU_R5VSTATUS - 5V Detector Status 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																											1	0	0	0	0	0
Access																											R	R	R	R	R	R
Name																											COLDSTART	LDODROPOUTDET	VBUSGTVREGI	VREGODET	VBUSDET	VREGIDET

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	COLDSTART	1	R	**Indicates If the Regulator is Going Through a Cold Start**
	When high the regulator is going through a cold start process.			
4	LDODROPOUTDET	0	R	**Regulator Dropout Detection**
	The selected input to the regulator is below target.			
3	VBUSGTVREGI	0	R	**Output of the Supply Comparator Between VBUS and VREGI**
	Output of the supply comparator between VBUS and VREGI.			
2	VREGODET	0	R	**VREGO Detected**
	Indicates presence of 5V regulator output.			
1	VBUSDET	0	R	**USB VBUS Detected**
	USB VBUS detected.			
0	VREGIDET	0	R	**VREGI Detected**
	Indicates presence of a battery on VREGI.			

9.5.40 EMU_R5VSYNC - 5V Read Status 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																																0
Access																																R
Name																																OUTLEVELBUSY

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	OUTLEVELBUSY	0	R	**5V Regulator Voltage Register Transfer Busy** Indicates the status of the 5V regulator Voltage Register transfer to the emu osc clock domain. Software cannot re-write the register until this signal goes down.

9.5.41 EMU_EM23PERNORETAINCMD - Clears Corresponding Bits in EM23PERNORETAINSTATUS Unlocking Access to Peripheral

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								0	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access								W1	W1		W1	W1		W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name								USBUNLOCK	RTCUNLOCK		ACMP2UNLOCK	ADC1UNLOCK		LETIMER1UNLOCK	LCDUNLOCK	LEUART1UNLOCK	LEUART0UNLOCK	CSENUNLOCK	LESENSE0UNLOCK	WDOG1UNLOCK	WDOG0UNLOCK	LETIMER0UNLOCK	ADC0UNLOCK	IDAC0UNLOCK	DAC0UNLOCK	I2C1UNLOCK	I2C0UNLOCK	PCNT2UNLOCK	PCNT1UNLOCK	PCNT0UNLOCK	ACMP1UNLOCK	ACMP0UNLOCK

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	USBUNLOCK	0	W1	**Clears Status Bit of USB and Unlocks Access to It**
	clears status bit of USB and unlocks access to it			
23	RTCUNLOCK	0	W1	**Clears Status Bit of RTC and Unlocks Access to It**
	clears status bit of RTC and unlocks access to it			
22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	ACMP2UNLOCK	0	W1	**Clears Status Bit of ACMP2 and Unlocks Access to It**
	clears status bit of ACMP2 and unlocks access to it			
20	ADC1UNLOCK	0	W1	**Clears Status Bit of ADC1 and Unlocks Access to It**
	clears status bit of ADC1 and unlocks access to it			
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	LETIMER1UNLOCK	0	W1	**Clears Status Bit of LETIMER1 and Unlocks Access to It**
	clears status bit of LETIMER1 and unlocks access to it			
17	LCDUNLOCK	0	W1	**Clears Status Bit of LCD and Unlocks Access to It**
	clears status bit of LCD and unlocks access to it			
16	LEUART1UNLOCK	0	W1	**Clears Status Bit of LEUART1 and Unlocks Access to It**
	clears status bit of LEUART1 and unlocks access to it			
15	LEUART0UNLOCK	0	W1	**Clears Status Bit of LEUART0 and Unlocks Access to It**
	clears status bit of LEUART0 and unlocks access to it			
14	CSENUNLOCK	0	W1	**Clears Status Bit of CSEN and Unlocks Access to It**
	clears status bit of CSEN and unlocks access to it			
13	LESENSE0UNLOCK	0	W1	**Clears Status Bit of LESENSE0 and Unlocks Access to It**
	clears status bit of LESENSE0 and unlocks access to it			

Bit	Name	Reset	Access	Description
12	WDOG1UNLOCK	0	W1	**Clears Status Bit of WDOG1 and Unlocks Access to It**
	clears status bit of WDOG1 and unlocks access to it			
11	WDOG0UNLOCK	0	W1	**Clears Status Bit of WDOG0 and Unlocks Access to It**
	clears status bit of WDOG0 and unlocks access to it			
10	LETIMER0UNLOCK	0	W1	**Clears Status Bit of LETIMER0 and Unlocks Access to It**
	clears status bit of LETIMER0 and unlocks access to it			
9	ADC0UNLOCK	0	W1	**Clears Status Bit of ADC0 and Unlocks Access to It**
	clears status bit of ADC0 and unlocks access to it			
8	IDAC0UNLOCK	0	W1	**Clears Status Bit of IDAC0 and Unlocks Access to It**
	clears status bit of IDAC0 and unlocks access to it			
7	DAC0UNLOCK	0	W1	**Clears Status Bit of DAC0 and Unlocks Access to It**
	clears status bit of DAC0 and unlocks access to it			
6	I2C1UNLOCK	0	W1	**Clears Status Bit of I2C1 and Unlocks Access to It**
	clears status bit of I2C1 and unlocks access to it			
5	I2C0UNLOCK	0	W1	**Clears Status Bit of I2C0 and Unlocks Access to It**
	clears status bit of I2C0 and unlocks access to it			
4	PCNT2UNLOCK	0	W1	**Clears Status Bit of PCNT2 and Unlocks Access to It**
	clears status bit of PCNT2 and unlocks access to it			
3	PCNT1UNLOCK	0	W1	**Clears Status Bit of PCNT1 and Unlocks Access to It**
	clears status bit of PCNT1 and unlocks access to it			
2	PCNT0UNLOCK	0	W1	**Clears Status Bit of PCNT0 and Unlocks Access to It**
	clears status bit of PCNT0 and unlocks access to it			
1	ACMP1UNLOCK	0	W1	**Clears Status Bit of ACMP1 and Unlocks Access to It**
	clears status bit of ACMP1 and unlocks access to it			
0	ACMP0UNLOCK	0	W1	**Clears Status Bit of ACMP0 and Unlocks Access to It**
	clears status bit of ACMP0 and unlocks access to it			

9.5.42 EMU_EM23PERNORETAINSTATUS - Status Indicating If Peripherals Were Powered Down in EM23, Subsequently Locking Access to It

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								0	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access								R	R		R	R		R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name								USBLOCKED	RTCLOCKED		ACMP2LOCKED	ADC1LOCKED		LETIMER1LOCKED	LCDLOCKED	LEUART1LOCKED	LEUART0LOCKED	CSENLOCKED	LESENSE0LOCKED	WDOG1LOCKED	WDOG0LOCKED	LETIMER0LOCKED	ADC0LOCKED	IDAC0LOCKED	DAC0LOCKED	I2C1LOCKED	I2C0LOCKED	PCNT2LOCKED	PCNT1LOCKED	PCNT0LOCKED	ACMP1LOCKED	ACMP0LOCKED

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	USBLOCKED	0	R	**Indicates If USB Powered Down During EM23**
	Indicates if USB powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
23	RTCLOCKED	0	R	**Indicates If RTC Powered Down During EM23**
	Indicates if RTC powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	ACMP2LOCKED	0	R	**Indicates If ACMP2 Powered Down During EM23**
	Indicates if ACMP2 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
20	ADC1LOCKED	0	R	**Indicates If ADC1 Powered Down During EM23**
	Indicates if ADC1 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	LETIMER1LOCKED	0	R	**Indicates If LETIMER1 Powered Down During EM23**
	Indicates if LETIMER1 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
17	LCDLOCKED	0	R	**Indicates If LCD Powered Down During EM23**
	Indicates if LCD powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
16	LEUART1LOCKED	0	R	**Indicates If LEUART1 Powered Down During EM23**
	Indicates if LEUART1 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
15	LEUART0LOCKED	0	R	**Indicates If LEUART0 Powered Down During EM23**
	Indicates if LEUART0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			

Bit	Name	Reset	Access	Description
14	CSENLOCKED	0	R	**Indicates If CSEN Powered Down During EM23**
	Indicates if CSEN powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
13	LESENSE0LOCKED	0	R	**Indicates If LESENSE0 Powered Down During EM23**
	Indicates if LESENSE0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
12	WDOG1LOCKED	0	R	**Indicates If WDOG1 Powered Down During EM23**
	Indicates if WDOG1 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
11	WDOG0LOCKED	0	R	**Indicates If WDOG0 Powered Down During EM23**
	Indicates if WDOG0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
10	LETIMER0LOCKED	0	R	**Indicates If LETIMER0 Powered Down During EM23**
	Indicates if LETIMER0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
9	ADC0LOCKED	0	R	**Indicates If ADC0 Powered Down During EM23**
	Indicates if ADC0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
8	IDAC0LOCKED	0	R	**Indicates If IDAC0 Powered Down During EM23**
	Indicates if IDAC0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
7	DAC0LOCKED	0	R	**Indicates If DAC0 Powered Down During EM23**
	Indicates if DAC0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
6	I2C1LOCKED	0	R	**Indicates If I2C1 Powered Down During EM23**
	Indicates if I2C1 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
5	I2C0LOCKED	0	R	**Indicates If I2C0 Powered Down During EM23**
	Indicates if I2C0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
4	PCNT2LOCKED	0	R	**Indicates If PCNT2 Powered Down During EM23**
	Indicates if PCNT2 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
3	PCNT1LOCKED	0	R	**Indicates If PCNT1 Powered Down During EM23**
	Indicates if PCNT1 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
2	PCNT0LOCKED	0	R	**Indicates If PCNT0 Powered Down During EM23**
	Indicates if PCNT0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			
1	ACMP1LOCKED	0	R	**Indicates If ACMP1 Powered Down During EM23**
	Indicates if ACMP1 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			

Bit	Name	Reset	Access	Description
0	ACMP0LOCKED	0	R	**Indicates If ACMP0 Powered Down During EM23**
	Indicates if ACMP0 powered down during EM23. Access to this peripheral locked until this bit cleared using EM23PERNORETAINCMD			

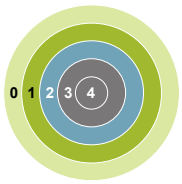
9.5.43 EMU_EM23PERNORETAINCTRL - When Set Corresponding Peripherals May Get Powered Down in EM23

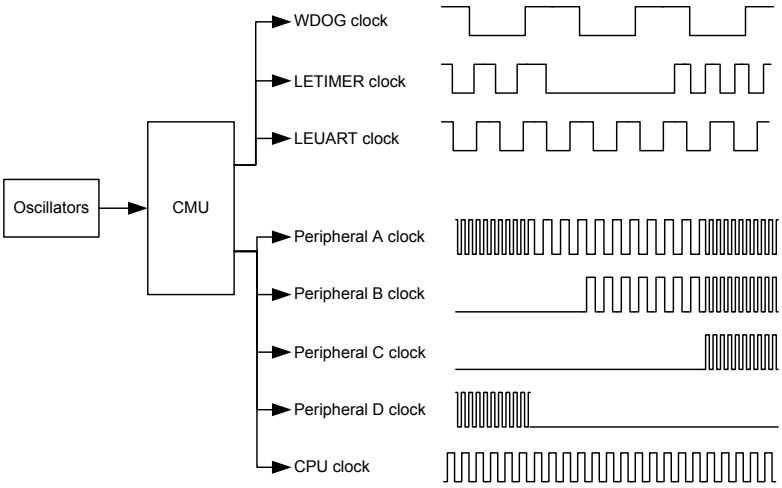
Offset	Bit Position																													
0x108	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset								0	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access								RW	RW		RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name								USBDIS	RTCDIS		ACMP2DIS	ADC1DIS		LETIMER1DIS	LCDDIS	LEUART1DIS	LEUART0DIS	CSENDIS	LESENSE0DIS	WDOG1DIS	WDOG0DIS	LETIMER0DIS	ADC0DIS	IDAC0DIS	VDAC0DIS	I2C1DIS	I2C0DIS	PCNT2DIS	PCNT1DIS	PCNT0DIS	ACMP1DIS	ACMP0DIS

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	USBDIS	0	RW	**Allow Power Down of USB During EM23**
	Allow power down of USB during EM23			
23	RTCDIS	0	RW	**Allow Power Down of RTC During EM23**
	Allow power down of RTC during EM23			
22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	ACMP2DIS	0	RW	**Allow Power Down of ACMP2 During EM23**
	Allow power down of ACMP2 during EM23			
20	ADC1DIS	0	RW	**Allow Power Down of ADC1 During EM23**
	Allow power down of ADC1 during EM23			
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	LETIMER1DIS	0	RW	**Allow Power Down of LETIMER1 During EM23**
	Allow power down of LETIMER1 during EM23			
17	LCDDIS	0	RW	**Allow Power Down of LCD During EM23**
	Allow power down of LCD during EM23			
16	LEUART1DIS	0	RW	**Allow Power Down of LEUART1 During EM23**
	Allow power down of LEUART1 during EM23			
15	LEUART0DIS	0	RW	**Allow Power Down of LEUART0 During EM23**
	Allow power down of LEUART0 during EM23			
14	CSENDIS	0	RW	**Allow Power Down of CSEN During EM23**
	Allow power down of CSEN during EM23			
13	LESENSE0DIS	0	RW	**Allow Power Down of LESENSE0 During EM23**
	Allow power down of LESENSE0 during EM23			
12	WDOG1DIS	0	RW	**Allow Power Down of WDOG1 During EM23**
	Allow power down of WDOG1 during EM23			

Bit	Name	Reset	Access	Description
11	WDOG0DIS	0	RW	**Allow Power Down of WDOG0 During EM23**
	Allow power down of WDOG0 during EM23			
10	LETIMER0DIS	0	RW	**Allow Power Down of LETIMER0 During EM23**
	Allow power down of LETIMER0 during EM23			
9	ADC0DIS	0	RW	**Allow Power Down of ADC0 During EM23**
	Allow power down of ADC0 during EM23			
8	IDAC0DIS	0	RW	**Allow Power Down of IDAC0 During EM23**
	Allow power down of IDAC0 during EM23			
7	VDAC0DIS	0	RW	**Allow Power Down of DAC0 During EM23**
	Allow power down of DAC0 during EM23			
6	I2C1DIS	0	RW	**Allow Power Down of I2C1 During EM23**
	Allow power down of I2C1 during EM23			
5	I2C0DIS	0	RW	**Allow Power Down of I2C0 During EM23**
	Allow power down of I2C0 during EM23			
4	PCNT2DIS	0	RW	**Allow Power Down of PCNT2 During EM23**
	Allow power down of PCNT2 during EM23			
3	PCNT1DIS	0	RW	**Allow Power Down of PCNT1 During EM23**
	Allow power down of PCNT1 during EM23			
2	PCNT0DIS	0	RW	**Allow Power Down of PCNT0 During EM23**
	Allow power down of PCNT0 during EM23			
1	ACMP1DIS	0	RW	**Allow Power Down of ACMP1 During EM23**
	Allow power down of ACMP1 during EM23			
0	ACMP0DIS	0	RW	**Allow Power Down of ACMP0 During EM23**
	Allow power down of ACMP0 during EM23			

10. CMU - Clock Management Unit





Quick Facts

What?

The CMU controls oscillators and clocks. EFM32 Giant Gecko 12 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 an MCU. Low power oscillators combined with a flexible clock control scheme make it possible to minimize the energy consumption in any given application.

How?

The CMU can configure different clock sources, enable/disable clocks to peripherals on an individual basis and set the prescaler for the different 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 Hibernate/Shutoff) very efficient. The calibration feature ensures high accuracy RC oscillators. Several interrupts are available to avoid CPU polling of flags.

10.1 Introduction

The Clock Management Unit (CMU) is responsible for controlling the oscillators and clocks in the EFM32 Giant Gecko 12. The CMU provides the capability to turn on and off the clock on an individual basis to all peripheral modules in addition to enable/disable and configure the available oscillators. The high degree of flexibility enables software to minimize energy consumption in any specific application by not wasting power on peripherals and oscillators that do not need to be active.

10.2 Features

- Multiple clock sources available:
  - 4 MHz - 50 MHz High Frequency Crystal Oscillator (HFXO)
  - 1 MHz - 72 MHz High Frequency RC Oscillator (HFRCO)
  - 1 MHz - 50 MHz Auxiliary High Frequency RC Oscillator (AUXHFRCO)
  - 1 MHz - 50 MHz Universal High Frequency RC Oscillator (USHFRCO)
  - 32768 Hz Low Frequency Crystal Oscillator (LFXO)
  - 32768 Hz Low Frequency RC Oscillator (LFRCO)
  - 1000 Hz Ultra Low Frequency RC Oscillator (ULFRCO)
- All oscillator sources are low power.
- Fast start-up times.
- Spectrum-Spreading Digital Phase-Locked Loop.
- Separate prescalers for High Frequency Core Clocks (HFCORECLK), Bus Clocks (HFBUSCLK), and Peripheral Clocks (HFPERCLK, HFPERBCLK, HFPERCCLK).
- Individual clock prescaler selection for each Low Energy Peripheral.
- Clock gating on an individual basis to core modules and all peripherals.
- Selectable clock output to external pins and/or PRS.
- Wakeup interrupt for LFRCO or LFXO ready allows entry into EM2 DeepSleep while waiting for low-frequency oscillator startup. This avoids the need for software polling and saves power during oscillator startup.
- Auxiliary 1 MHz - 50 MHz RC oscillator (AUXHFRCO), which is asynchronous to the HFSRCCLK system clock, can be selected for ADC operation, LESENSE timing, SDIO operation, QSPI operation and debug trace.
- Universal 1 MHz - 50 MHz RC oscillator (USHFRCO), which can be selected for SDIO operation, QSPI operation, USB operation, and HFSRCCLK system clock.

10.3 Functional Description

An overview of the high frequency portion of the CMU is shown in Figure 10.1 CMU Overview - High Frequency Portion on page 362. An overview of the low frequency portion is shown in Figure 10.2 CMU Overview - Low Frequency Portion on page 363. These figures show the CMU for the largest device in the EFM32 family. Refer to the Configuration Summary in the device data sheet to see which core, and peripheral modules, and therefore clock connections, are present in a specific device.

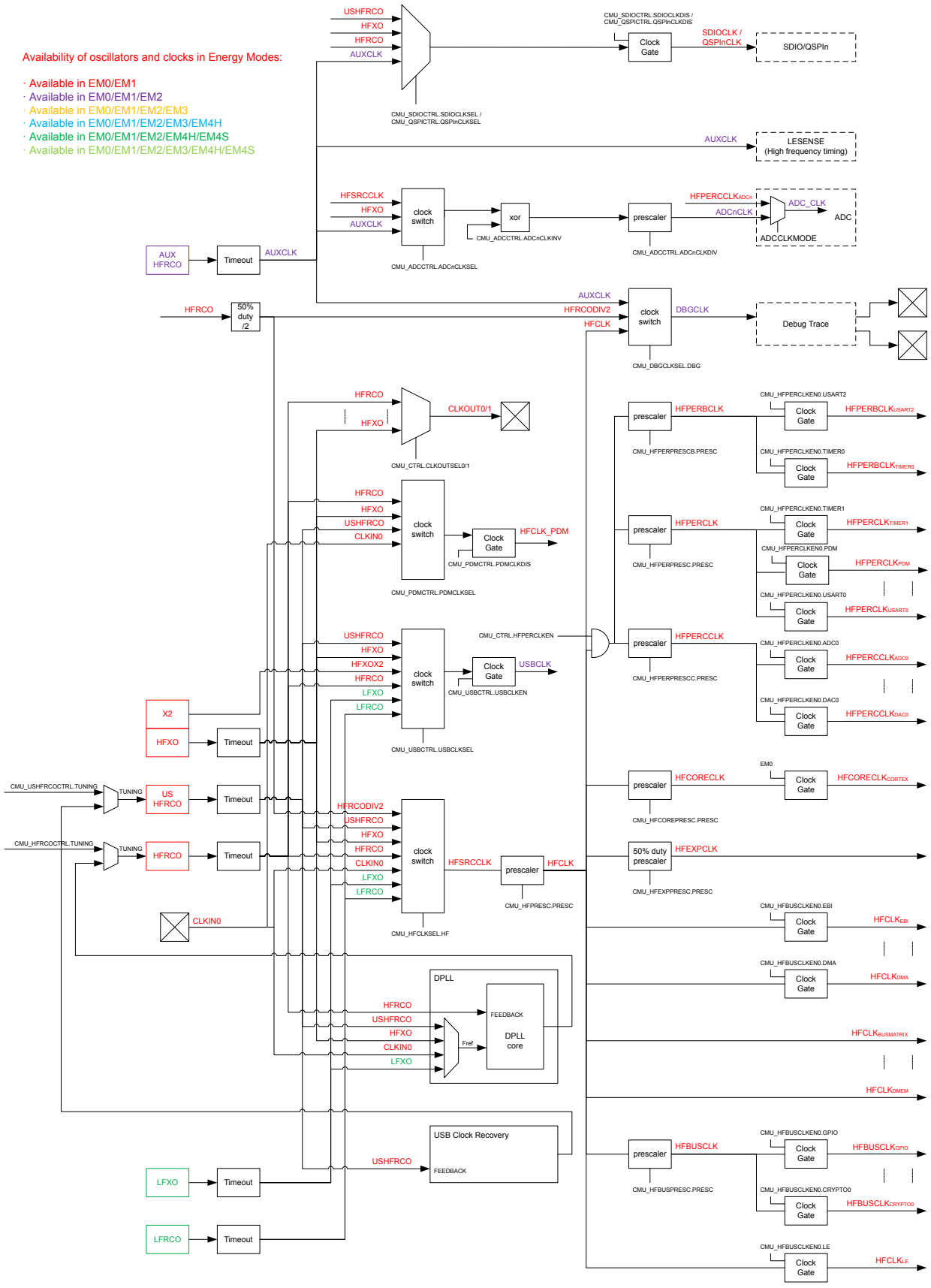


Figure 10.1. CMU Overview - High Frequency Portion

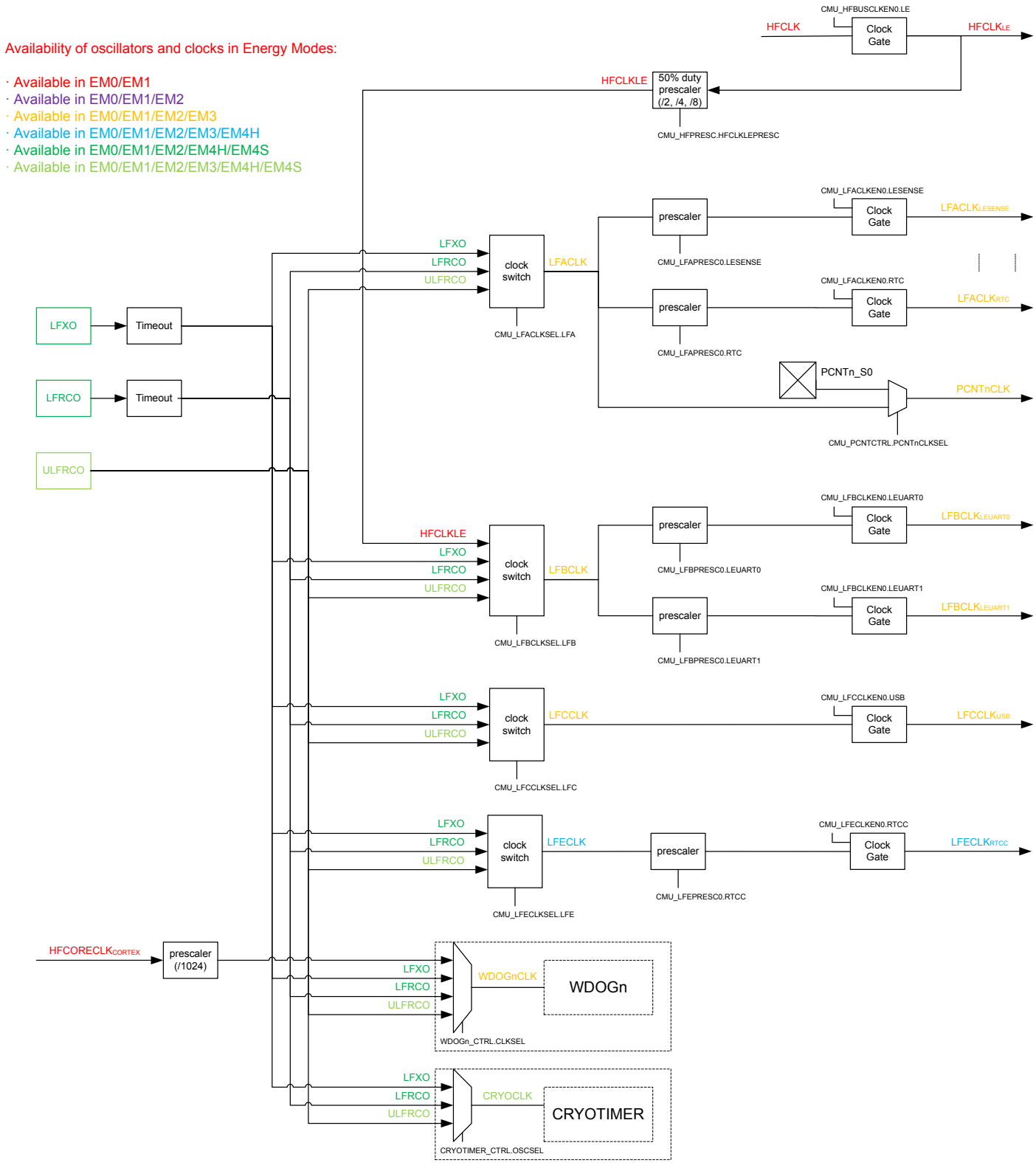


Figure 10.2. CMU Overview - Low Frequency Portion

10.3.1 System Clocks

Available system clock sources are detailed in the following sections.

10.3.1.1 HFCLK - High Frequency Clock

HFSRCCLK is the selected High Frequency Source Clock. HFCLK is an optionally prescaled version of HFSRCCLK. The HFSRCCLK, and therefore HFCLK, can be driven by a high-frequency oscillator, such as HFRCO, USHFRCO, HFRCODIV2 (see DPLL 10.3.12.1 Enabling and Disabling) or HFXO, or one of the low-frequency oscillators (LFRCO or LFXO). Additionally, HFSRCCLK can also be driven from a pin (CLKIN0) described in 10.3.6 Clock Input From a Pin. By default the HFRCO is selected. In most applications, one of the high frequency oscillators will be the preferred choice. To change the selected clock source, write to the HF bitfield in CMU_HFCLKSEL. The high frequency clock source can also be changed automatically by hardware as explained in 10.3.2.4.1 Automatic HFXO Start. The currently selected source for HFSRCCLK and HFCLK can be read from CMU_HFCLKSTATUS. The HFSRCCLK is running in EM0 Active and EM1 Sleep and is automatically stopped in EM2 DeepSleep. During Voltage Scaling (see 9.3.9 Voltage Scaling), if a fixed frequency oscillator source (i.e. HFXO or CLKIN0) exceeds the maximum system frequency supported, it must be disabled or not selected. Likewise, an adjustable oscillator source (i.e. HFRCO, USHFRCO or AUXHFRCO) must be configured to not exceed the maximum system frequency supported before voltage scaling is applied.

Note: If a low frequency clock (i.e. LFRCO or LFXO) is selected as source clock for HFSRCCLK via the HF bitfield in CMU_HFCLKSEL, then no register reads should be performed from Low Energy Peripherals for registers which can change value every clock cycle (e.g., a counter register). In addition to the peripherals on LFACLK, LFBCLK, LFCCLK and LFECLK, this restriction applies in general to any low frequency peripheral, which is not directly or indirectly clocked from HFSRCCLK (e.g., WDOGn).

HFCLK can optionally be prescaled by setting PRESC in CMU_HFPRESC to a non-zero value. This prescales HFCLK to all high frequency components and is typically used to save energy in applications where the system is not required to run at the highest frequency. The prescaler setting can be changed dynamically and the new setting takes effect immediately. HFCLK is used by the CMU and drives the prescalers that generate HFCORECLK, HFBUSCLK and HFPERCLK, HFPERBCLK, HFPERCCLK allowing for flexible clock prescaling. HFCLK is used for Bus and Memory System modules as for example the Bus Matrix, MSC and DMEM. HFCLK is also used to drive the bus interface to the Low Energy Peripherals as described further in 10.3.1.10 LFACLK - Low Frequency a Clock, 10.3.1.11 LFBCLK - Low Frequency B Clock and 10.3.1.13 LFECLK - Low Frequency E Clock. Some of the modules that are driven by HFCLK can be clock gated completely when not in use. This is done by clearing the clock enable bit for the specific module in CMU_HFBUSCLKEN0. Table 10.1 Clock Domain (HFCLK, HFBUSCLK) Per Peripheral on page 365 shows which peripherals are in the HFCLK domain.

10.3.1.2 HFCORECLK - High Frequency Core Clock

HFCORECLK is a prescaled version of HFCLK. This clock drives the Core Modules, which consists of the CPU and modules that are tightly coupled to the CPU (e.g., the cache). The prescale factor for prescaling HFCLK into HFCORECLK is set using the CMU_HFCOREPRESC register. The setting can be changed dynamically and the new setting takes effect immediately.

Note: If HFPERCLK, HFPERBCLK, HFPERCCLK runs faster than HFCORECLK, the number of clock cycles for each bus-access to peripheral modules will increase with the ratio between the clocks. Refer to 4.2.5 Bus Matrix for more details.

10.3.1.3 HFBUSCLK - High Frequency Bus Clock

HFBUSCLK is a prescaled version of HFCLK. HFBUSCLK is used to drive modules such as GPIO and GPCRC. The prescale factor for prescaling HFCLK into HFBUSCLK is set using the CMU_HFBUSPRESC register. The setting can be changed dynamically and the new setting takes effect immediately. Some of the modules that are driven by HFBUSCLK can be clock gated completely when not in use. This is done by clearing the clock enable bit for the specific module in CMU_HFBUSCLKEN0.

Table 10.1 Clock Domain (HFCLK, HFBUSCLK) Per Peripheral on page 365 shows which peripheral is in what clock domain (HFCLK or HFBUSCLK).

Table 10.1. Clock Domain (HFCLK, HFBUSCLK) Per Peripheral

Peripheral	Bus Clock
LE	HFCLK
EBI	HFCLK
SDIO	HFCLK
PRS	HFCLK
LDMA	HFCLK
USB	HFCLK
MSC	HFCLK
SMU	HFCLK
CRYPTO0	HFBUSCLK
GPIO	HFBUSCLK
GPCRC	HFBUSCLK
QSPI0	HFBUSCLK

10.3.1.4 HFPERCLK, HFPERBCLK, HFPERCCLK - High Frequency Peripheral Clocks

Like HFCORECLK, also HFPERCLK, HFPERBCLK, and HFPERCCLK are prescaled versions of HFCLK. These clocks drive the High-Frequency Peripherals. All the peripherals that are driven by these clocks can be clock gated individually when not in use. This is done by clearing the clock enable bit for the specific peripheral in CMU_HFPERCLKEN0 or CMU_HFPERCLKEN1. All high frequency peripheral clocks can be universally and simultaneously gated by clearing the HFPERCLKEN bit in the CMU_CTRL register. The prescale factors for prescaling HFCLK into HFPERCLK, HFPERBCLK, and HFPERCCLK are set using the CMU_HFPERPRESC, CMU_HFPERPRESCB, and CMU_HFPERPRESCC registers respectively. The setting can be changed dynamically and the new setting takes effect immediately.

Table 10.2 Peripheral Clock Domains (HFPERCLK, HFPERBCLK, HFPERCCLK) Per Peripheral on page 366 shows which peripheral is in what peripheral clock domain.

Table 10.2. Peripheral Clock Domains (HFPERCLK, HFPERBCLK, HFPERCCLK) Per Peripheral

Peripheral	Peripheral Clock
TIMER1	HFPERCLK
TIMER2	HFPERCLK
TIMER3	HFPERCLK
WTIMER0	HFPERCLK
WTIMER1	HFPERCLK
USART0	HFPERCLK
USART1	HFPERCLK
USART3	HFPERCLK
USART4	HFPERCLK
UART0	HFPERCLK
UART1	HFPERCLK
CAN0	HFPERCLK
CAN1	HFPERCLK
TRNG0	HFPERCLK
PDM	HFPERCLK
TIMER0	HFPERBCLK
USART2	HFPERBCLK
ACMP0	HFPERCCLK
ACMP1	HFPERCCLK
ACMP2	HFPERCCLK
I2C0	HFPERCCLK
I2C1	HFPERCCLK
ADC0	HFPERCCLK
ADC1	HFPERCCLK
CRYOTIMER	HFPERCCLK
VDAC0	HFPERCCLK
IDAC0	HFPERCCLK
CSEN	HFPERCCLK

Note: If HFPERCLK, HFPERBCLK or HFPERCCLK runs faster than HFCORECLK, the number of clock cycles for each bus-access to peripheral modules will increase with the ratio between the clocks. E.g. if a bus-access normally takes three cycles, it will take 9 cycles of HFCORECLK if HFPERCLK runs three times as fast as HFCORECLK.

10.3.1.5 ADCnCLK - ADC Core Clock

ADCnCLK is a selectable core clock for ADCn. There are three selectable sources for ADCnCLK: HFSRCCLK, HFXO and AUXHFRCO. In addition, the ADCnCLK can be disabled, which is the default setting. The selection is configured using the ADCnCLKSEL field in CMU_ADCCTRL. The ADCnCLKINV bit in CMU_ADCCTRL can be used to invert ADCnCLK. The ADCnCLKDIV bitfield in CMU_ADCCTRL can be used to prescale ADCnCLK. The bus interface of ADCn is clocked with HFBUSCLK.

10.3.1.6 USBCLK - USB Core Clock

USBCLK is the selected clock for USB. There are six selectable sources for USBCLK: USHFRCO, HFXO, HFXOX2, HFRCO, LFXO and LFRCO. The selection is configured using the USBCLKSEL field in CMU_USBCTRL. The USBCLKDIS in CMU_USBCTRL can be used to gate off USBCLK. The bus interface of USB is clocked with HFBUSCLK.

10.3.1.7 QSPInCLK - QSPI Reference Clock

QSPInCLK is the selected reference clock for QSPIn. There are four selectable sources for QSPInCLK: HFRCO, HFXO, AUXHFRCO and USHFRCO. The selection is configured using the QSPInCLKSEL field in CMU_QSPICTRL. The QSPInCLKDIS in CMU_QSPICTRL can be used to gate off QSPInCLK. The QSPInCLKENS bit in CMU_STATUS indicates whether the QSPInCLK has been enabled or disabled. The clock selection should only be changed when QSPInCLK is disabled. The bus interface of QSPIn is clocked with $HFBUSCLK_{QSPIn}$.

Note: The user should disable QSPInCLK and wait for it to be disabled (QSPInCLKENS = 0) before entering EM2 DeepSleep or EM3 Stop.

10.3.1.8 SDIOCLK - SDIO Reference Clock

SDIOCLK is the selected reference clock for the SDIO. There are four selectable sources for SDIOCLK: HFRCO, HFXO, AUXHFRCO and USHFRCO. The selection is configured using the SDIOCLKSEL field in CMU_SDIOCTRL. The SDIOCLKDIS in CMU_SDIOCTRL can be used to gate off SDIOCLK. The SDIOCLKENS bit in CMU_STATUS indicates whether the SDIOCLK has been (completely) enabled or disabled. The clock selection should only be changed when SDIOCLK is disabled. The bus interface of SDIO is clocked with $HFCLK_{SDIO}$.

Note:

- The user should disable SDIOCLK and wait for it to be disabled (SDIOCLKENS = 0) before entering EM2 DeepSleep or EM3 Stop.
- The user should disable SDIOCLK and wait for it to be disabled (SDIOCLKENS = 0) before voltage scaling down from VSCALE2.

10.3.1.9 PDMCLK - PDM Core Clock

PDMCLK is the selected core clock for the PDM. There are four selectable sources for PDMCLK: HFRCO, HFXO, USHFRCO and from pin CLKIN0. The selection is configured using the PDMCLKSEL field in CMU_PDMCTRL. The PDMCLKEN in CMU_PDMCTRL can be used to gate off PDMCLK. The PDMCLKENS bit in CMU_STATUS indicates whether the PDMCLK has been enabled or disabled. The clock selection should only be changed when PDMCLK is disabled.

10.3.1.10 LFACLK - Low Frequency a Clock

LFACLK is the selected clock for the Low Energy A Peripherals. There are several selectable sources for LFACLK: LFRCO, LFXO and ULFRCO. In addition, the LFACLK can be disabled, which is the default setting. The selection is configured using the LFA field in CMU_LFACLKSEL.

The bus interface to the Low Energy A Peripherals is clocked by $HFCLK_{LE}$ and this clock therefore needs to be enabled when programming a Low Energy (LE) peripheral.

Each Low Energy Peripheral that is clocked by LFACLK has its own prescaler setting and enable bit. The prescaler settings are configured using CMU_LFAPRESC0 and the clock enable bits can be found in CMU_LFACLKEN0.

When operating in oversampling mode, the pulse counters are clocked by LFACLK. This is configured for each pulse counter (n) individually by setting PCNTnCLKSEL in CMU_PCNTCTRL.

10.3.1.11 LFBCLK - Low Frequency B Clock

LFBCLK is the selected clock for the Low Energy B Peripherals. There are several selectable sources for LFBCLK: LFRCO, LFXO, HFCLKLE and ULFRCO. In addition, the LFBCLK can be disabled, which is the default setting. The selection is configured using the LFB field in CMU_LFBCLKSEL. The HFCLKLE setting allows the Low Energy B Peripherals to be used as high-frequency peripherals.

The bus interface to the Low Energy B Peripherals is clocked by $HFCLK_{LE}$ and this clock therefore needs to be enabled when programming a LE peripheral.

Note: If HFCLKLE is selected as LFBCLK, the clock will stop in EM2 DeepSleep and EM3 Stop.

Each Low Energy Peripheral that is clocked by LFBCLK has its own prescaler setting and enable bit. The prescaler settings are configured using CMU_LFBPRESC0 and the clock enable bits can be found in CMU_LFBCLKEN0.

10.3.1.12 LFCCLK - Low Frequency C Clock

LFCCLK is the selected clock for the Low Energy C Peripherals. There are three selectable sources for LFCCLK: LFRCO, LFXO and ULFRCO. In addition, the LFCCLK can be disabled, which is the default setting. The selection is configured using the LFC field in CMU_LFCCLKSEL. The related clock enable bits can be found in CMU_LFCCLKEN0.

10.3.1.13 LFECLK - Low Frequency E Clock

LFECLK is the selected clock for the Low Energy E Peripherals. There are several selectable sources for LFECLK: LFRCO, LFXO and ULFRCO. In addition, the LFECLK can be disabled, which is the default setting. The selection is configured using the LFE field in CMU_LFECLKSEL.

The bus interface to the Low Energy E Peripherals is clocked by $HFCLK_{LE}$ and this clock therefore needs to be enabled when programming a LE peripheral.

Note: LFECLK is in a different power domain than LFACLK, LFBCLK and LFCCLK, which makes it available all the way down to EM4 Hibernate.

Each Low Energy Peripheral that is clocked by LFECLK has its own prescaler setting and enable bit. The prescaler settings are configured using CMU_LFEPRESC0 and the clock enable bits can be found in CMU_LFECLKEN0.

10.3.1.14 PCNTnCLK - Pulse Counter N Clock

Each available pulse counter is driven by its own clock, PCNTnCLK where n is the pulse counter instance number. Each pulse counter can be configured to use an external pin (PCNTn_S0) or LFACLK as PCNTnCLK.

10.3.1.15 WDOGnCLK - Watchdog Timer Clock

The Watchdog Timer (WDOGn) can be configured to use one of many different clock sources. Refer to CLKSEL field in WDOGn_CTRL for a complete list.

10.3.1.16 CRYOCLK - CRYOTIMER Clock

The CRYOTIMER clock can be configured to use one of many different clock sources. Refer to OSCSEL field in CRYOTIMER_CTRL for a complete list. The CRYOTIMER can also run in EM4 Hibernate/Shutoff provided that its selected clock is kept enabled as configured in EMU_EM4CTRL.

10.3.1.17 AUXCLK - Auxiliary Clock

AUXCLK is a 1 MHz - 50 MHz clock driven by a separate RC oscillator, the AUXHFRCO. This clock can be used for ADC operation LESENSE operation. When the AUXHFRCO is selected as the ADCn clock via the ADCnCLKSEL bitfield in the CMU_ADCCTRL register, or if needed by LESENSE, this clock will become active automatically when needed. Even if the AUXHFRCO has not been enabled explicitly by software, the ADC or LESENSE can automatically start and stop it. The AUXHFRCO is explicitly enabled by writing a 1 to AUXHFRCOEN in CMU_OSCENCMD. This explicit enabling is required when selecting the AUXCLK for SDIO operation, QSPI operation or SWO operation.

10.3.1.18 Debug Trace Clock

The CMU selects the clock used for debug trace via the DBGCLKSEL register. The user can use HFRCODIV2, AUXHFRCO or the HFCLK. The selected debug trace clock will be used to run the Cortex-M4 trace logic.

Note: When using AUXHFRCO as the debug trace clock, it must be stopped before entering EM2 or EM3.

10.3.2 Oscillators

Control of the various oscillators available in the device is detailed in the following sections.

10.3.2.1 Enabling and Disabling

The different oscillators can typically be enabled and disabled via both hardware and software mechanisms. Enabling via software is done by setting the corresponding enable bit in the CMU_OSCENCMD register. Disabling via software is done by setting the corresponding disable bit in CMU_OSCENCMD. Enabling via hardware can be performed by various peripherals and varies per oscillator. Disabling via hardware is typically performed on entry of low energy modes. The enable and disable mechanisms for each of the oscillators are summarized in Table 10.3 Software Based and Hardware Based Enabling and Disabling of Oscillators on page 369 and described in more detail below.

Table 10.3. Software Based and Hardware Based Enabling and Disabling of Oscillators

Oscillator	SW Enable	SW Disable	HW Enable	HW Disable
ULFRCO	-	-	Enabled when in EM0/EM1/EM2/EM3/EM4H.	EM4S entry depending on configuration in EMU_EM4CTRL.
LFRCO	Via LFRCOEN in CMU_OSCENCMD.	Via LFRCODIS in CMU_OSCENCMD.	Via WDOGn if it is configured to use LFRCO as its clock source via the CLKSEL bitfield in WDOGn_CTRL while SWOSCBLOCK is set.	EM3 entry. EM4 entry depending on configuration in EMU_EM4CTRL.
LFXO	Via LFXOEN in CMU_OSCENCMD.	Via LFXODIS in CMU_OSCENCMD.	Via WDOGn if it is configured to use LFXO as its clock source via the CLKSEL bitfield in WDOGn_CTRL while SWOSCBLOCK is set.	EM3 entry. EM4 entry depending on configuration in EMU_EM4CTRL.
HFRCO	Via HFRCOEN in CMU_OSCENCMD.	Via HFRCODIS in CMU_OSCENCMD.	Reset exit. EM2/EM3 exit. Automatic control by LEUART RX/TX DMA wake-up as configured in LEUARTn_CTRL.	EM2/EM3/EM4 entry. Automatic control by LEUART RX/TX DMA wake-up as configured in LEUARTn_CTRL. Automatic start and selection of HFXO causes HFRCO disable.
AUXHFRCO	Via AUXHFRCOEN in CMU_OSCENCMD.	Via AUXHFRCODIS in CMU_OSCENCMD.	Automatic control by ADC and LESENSE.	EM2/EM3/EM4 entry. Automatic control by ADC and LESENSE even in EM2/EM3.
USHFRCO	Via USHFRCOEN in CMU_OSCENCMD.	Via USHFRCODIS in CMU_OSCENCMD.	-	EM2/EM3/EM4 entry.
HFXO	Via HFXOEN in CMU_OSCENCMD.	Via HFXODIS in CMU_OSCENCMD.	Automatic start by EM0/EM1 entry as configured in CMU_HFXOCTRL.	EM2/EM3/EM4 entry.

10.3.2.1.1 LFRCO and LFXO

The LFXO and LFRCO can be enabled and disabled by software via the CMU_OSCENCMD register. WDOGn can be configured to force the LFXO or LFRCO to become (and remain) enabled when such an oscillator is selected as its clock source via the CLKSEL bitfield in the WDOGn_CTRL register while SWOSCBLOCK is set. In that case LFXODIS and LFRCODIS commands are blocked. They are automatically disabled when entering EM3. Upon EM4 entry they are default turned off, but they can optionally be retained depending on the EMU_EM4CTRL configuration. Retaining of the LFXO or LFRCO in EM4 is needed if such an oscillator is required by a specific peripheral in EM4. Retaining can also be used to guarantee quick oscillator availability after EM4 exit.

The oscillators should never be retained in case they are off before entering EM4. The following are the valid ways of using the LFXO/LFRCO retention mechanism:

- Turn on LFXO/LFRCO always (even in EM4):
  1. POR
  2. Enable LFXO/LFRCO
  3. Enable RETAINLFXO/RETAINLFRCO
  4. EM4 entry
  5. LFXO/LFRCO are retained and remain running in EM4
  6. EM4 wakeup
  7. Enable LFXO/LFRCO
  8. Set EM4UNLATCH in EMU_CMD
- Turn off LFXO/LFRCO in EM4:
  1. POR
  2. Disable RETAINLFXO/RETAINLFRCO (default)
  3. Enable LFXO/LFRCO
  4. EM4 entry
  5. LFXO/LFRCO are off in EM4
  6. EM4 wakeup
  7. Enable LFXO/LFRCO
  8. Set EM4UNLATCH in EMU_CMD
- Turn on LFXO/LFRCO after EM4 exit:
  1. POR
  2. Disable RETAINLFXO/RETAINLFRCO (default)
  3. Enable LFXO/LFRCO
  4. EM4 entry
  5. LFXO/LFRCO are off in EM4
  6. EM4 wakeup
  7. Enable LFXO/LFRCO
  8. Set EM4UNLATCH in EMU_CMD
  9. Enable RETAINLFXO/RETAINLFRCO

In summary RETAINLFXO/RETAINLFRCO should either be changed once after POR and kept static, or they can be changed on-the-fly only after asserting EM4UNLATCH.

Note:

- In order to support usage of LFRCO and LFXO in EM4, their settings are automatically latched upon EM4 entry. These settings remain latched upon wake-up from EM4 to EM0 although the related registers (CMU_LFRCOCTRL, CMU_LFXOCTRL, CMU_LFECLKSEL, CMU_LFECLKEN0 and CMU_LEEPRESC0) will have been reset. The registers can be rewritten by software, but they will only affect the LFRCO and LFXO after unlatching their settings by setting EM4UNLATCH in the EMU_CMD register.
- Turning off the LFRCO and LFXO upon EM4 Hibernate/Shutoff entry is most easily done by using the RETAINLFRCO and RETAINLFXO bitfields from the EMU_EM4CTRL register, which are default such that the LFRCO and LFXO are turned off automatically upon EM4 Hibernate/Shutoff entry. Alternatively the LFRCO and LFXO can be disabled via the CMU_OSCENCMD register, in which case software should wait for the oscillators to be properly disabled before executing the EM4 Hibernate/Shutoff entry routine.

After enabling the LFRCO (or LFXO), it should not be disabled before it has been signaled to be ready. Similarly, after disabling the LFRCO (or LFXO), it should not be re-enabled before it has been signaled to be non-ready. Before entering EM4, software should check that the LFRCO (or LFXO) is signaled to be ready before allowing or initiating the EM4 entry if that oscillator is required in EM4. Also, to guarantee latching the latest settings, no control write should be ongoing upon EM4 entry as can be checked via the CMU_SYNCBUSY register. Typical enable and disable sequences are as follows:

```
CMU->OSCENCMD = CMU_OSCENCMD_LFRCOEN;
while ((CMU->STATUS & CMU_STATUS_LFRCORDY) != CMU_STATUS_LFRCORDY);

CMU->OSCENCMD = CMU_OSCENCMD_LFRCODIS;
while ((CMU->STATUS & CMU_STATUS_LFRCORDY) == CMU_STATUS_LFRCORDY);
```

When the LFXO is disabled, the interface to the LFXTAL_N and LFXTAL_P pins are set in a high-Z state. The XTAL oscillations will not stop immediately when LFXO is disabled, but typically die out gradually over some 100 ms. If the LFXO is enabled before XTAL oscillations have had time to reach zero amplitude, startup time can be significantly shorter.

Note: The LFRCORDY and LFXORDY interrupts can be used to wake up the system from EM2 DeepSleep. In this way busy waiting for the LFRCO or LFXO to become ready can be avoided by going into EM2 after enabling these oscillators and sleeping until the interrupt causes a wakeup.

10.3.2.1.2 ULFRCO

The ULFRCO is automatically enabled in EM0, EM1, EM2, EM3, and EM4H and cannot be controlled via CMU_OSCENCMD. It is automatically disabled upon entering EM4S unless prevented by the configuration in EMU_EM4CTRL.

10.3.2.1.3 HFRCO

The HFRCO can be enabled and disabled by software via the CMU_OSCENCMD register. The HFRCO is disabled automatically when entering EM2, EM3, or EM4. Further hardware based enabling and disabling can be performed by the LEUART when using automatic RX/TX DMA wakeup as controlled by the RXDMAWU and TXDMAWU bits in the LEUARTn_CTRL register. An automatic start and selection of the HFXO will lead to an automatic HFRCO disabling. Since HFRCO also serves as the local oscillator for DPLL (10.3.12 Digital Phase-Locked Loop), it is enabled/disabled when DPLL is enabled/disabled.

The supported HFRCO frequency range is from 1 MHz to 72 MHz. The default HFRCO frequency is 19 MHz

10.3.2.1.4 USHFRCO

The USHFRCO can be enabled and disabled by software via the CMU_OSCENCMD register. The USHFRCO is disabled automatically when entering EM2, EM3, or EM4.

The supported USHFRCO frequency range is from 1 MHz to 50 MHz. The default USHFRCO frequency is 48 MHz

10.3.2.1.5 HFXO

The HFXO can be enabled and disabled by software via the CMU_OSCENCMD register. The HFXO is disabled automatically when entering EM2, EM3, or EM4. Hardware based HFXO enabling can be initiated by various peripherals as configured via theAUTOSTARTEM0EM1, and AUTOSTARTSELEM0EM1 bits in the CMU_HFXOCTRL register. The interaction between hardware based and software based control of the HFXO is further explained in 10.3.2.4.1 Automatic HFXO Start.

The supported HFXO frequency range is from 4 MHz to 50 MHz.

After enabling the HFXO, it should not be disabled before it has been signaled to be enabled. Similarly, after disabling the HFXO it should not be re-enabled before it has been signaled to be non-enabled. Typical enable and disable sequences are as follows:

```
CMU->OSCENCMD = CMU_OSCENCMD_HFXOEN;
while ((CMU->STATUS & CMU_STATUS_HFXOENS) != CMU_STATUS_HFXOENS);

CMU->OSCENCMD = CMU_OSCENCMD_HFXODIS;
while ((CMU->STATUS & CMU_STATUS_HFXOENS) == CMU_STATUS_HFXOENS);
```

10.3.2.1.6 AUXHFRCO

The AUXHFRCO can be enabled and disabled by software via the CMU_OSCENCMD register. The AUXHFRCO is disabled automatically when entering EM2, EM3, or EM4. Hardware based AUXHFRCO enabling and disabling is however performed by the ADC module when AUXCLK is selected for its operation and by the LESENSE module making it available even when being in EM2/EM3.

The supported AUXHFRCO frequency range is from 1 MHz to 50 MHz. The default AUXHFRCO frequency is 19 MHz

After enabling the AUXHFRCO, it should not be disabled before it has been signaled to be enabled. Similarly, after disabling the AUXHFRCO, it should not be re-enabled before it has been signaled to be non-enabled. Typical enable and disable sequences are as follows:

```
CMU->OSCENCMD = CMU_OSCENCMD_AUXHFRCOEN;
while ((CMU->STATUS & CMU_STATUS_AUXHFRCOENS) != CMU_STATUS_AUXHFRCOENS);

CMU->OSCENCMD = CMU_OSCENCMD_AUXHFRCODIS;
while ((CMU->STATUS & CMU_STATUS_AUXHFRCOENS) == CMU_STATUS_AUXHFRCOENS);
```

Note: When using AUXHFRCO as the debug trace clock (as selected in CMU_DBGCLKSEL), it must be stopped before entering EM2 or EM3.

10.3.2.2 Oscillator Start-up Time and Time-out

The start-up time differs per oscillator and the usage of an oscillator clock can further be delayed by a time-out. The LFRCO, LFXO and the HFXO have a configurable time-out which is set by software in the (various) TIMEOUT bitfields of the CMU_LFRCOCTRL, CMU_LFXOCTRL and CMU_HFXOTIMEOUTCTRL registers respectively. The time-out delays the assertion of the READY signal for LFRCO, LFXO and HFXO and should allow for enough time for the oscillator to stabilize. The time-out can be optimized for the chosen crystal (for LFXO and HFXO) used in the application. In case LFRCO and/or LFXO has been retained throughout EM4 Hibernate/Shut-off, such retained oscillators can be quickly restarted for use as LFACLK, LFBCLK, LFCCLK or LFECLK by using the minimum TIME-OUT settings for them. For the other RC oscillators (HFRCO, AUXHFRCO, USHFRCO, and ULFRCO), the start-up time is known and a fixed time-out is used.

There are individual bits in the CMU_STATUS register for each oscillator indicating the status of the oscillator:

- ENABLED - Indicates that the oscillator is enabled
- READY - Start-up time including time-out is exceeded

These status bits are located in the CMU_STATUS register.

Additionaly, the HFXO has a second time-out counter which can be used to achieve deterministic start-up time based on timing from the LFXO, ULFRCO, or LFRCO. This second counter runs off LFECLK and can be programmed via the LFTIMEOUT bitfield in the CMU_HFXOCTRL register. It can be used when waking up from EM2 when either ULFRCO, LFRCO or LFXO is already running and stable. In this case the HFXO ready assertion can be delayed with the number of LFECLK cycles as programmed in LFTIMEOUT. The HFXO ready signal is asserted when both the TIMEOUT counter (configured via the CMU_HFXOTIMEOUTCTRL register) and the LFTIMEOUT counter (configured via CMU_HFXOCTRL register) have timed out as shown in Figure 10.3 CMU Deterministic HFXO startup using LFTIMEOUT on page 373. The TIMEOUT should cover the actual crystal startup time. Typically the time base used for the TIMEOUT counter is not as accurate as the time base accuracy that can be achieved for the LFTIMEOUT counter, specifically if that one is based on the LFXO timing. If LFTIMEOUT is triggered before TIMEOUT is triggered, then the LFTIMEOUTERR bitfield in CMU_IF will be set to 1. Note that use of LFTIMEOUT requires that the peripheral causing the wake-up is on the LFECLK domain.

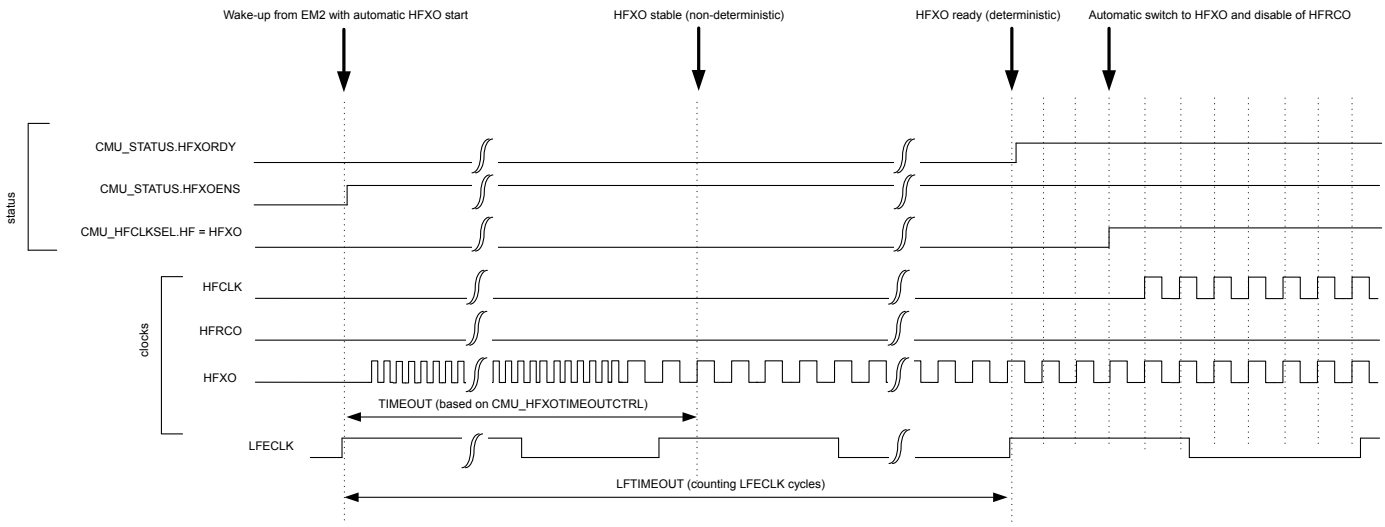


Figure 10.3. CMU Deterministic HFXO startup using LFTIMEOUT

The startup behavior of the HFXO also depends on how and how long the HFXO is disabled.

10.3.2.3 Switching Clock Source

The HFRCO oscillator is a low energy oscillator with extremely short start-up time. Therefore, this oscillator is always chosen by hardware as the clock source for HFCLK when the device starts up (e.g., after reset and after waking up from EM2 DeepSleep and EM3 Stop). After reset, the HFRCO frequency is 19 MHz.

Software can switch between the different clock sources at run-time. For example, when the HFRCO is the clock source, software can switch to HFXO by writing the field HF in the CMU_HFCLKSEL command register. See Figure 10.4 CMU Switching from HFRCO to HFXO before HFXO is ready on page 374 for a description of the sequence of events for this specific operation.

Note: Before switching the HFCLKSRC to HFXO via the HF bitfield in CMU_HFCLKSEL it is important to first enable the HFXO. Switching to a disabled oscillator will effectively stop HFSRCCLK and only a reset can recover the system.

When selecting an oscillator which has been enabled, but which is not ready yet, the HFSRCCLK will stop for the duration of the oscillator start-up time since the oscillator driving it is not ready. This effectively stalls the Core Modules and the High-Frequency Peripherals. 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) has timed out and provides a reliable clock. This sequence of events is shown in Figure 10.5 CMU Switching from HFRCO to HFXO after HFXO is ready on page 375.

A separate flag is set when the oscillator is ready. This flag can also be configured to generate an interrupt.

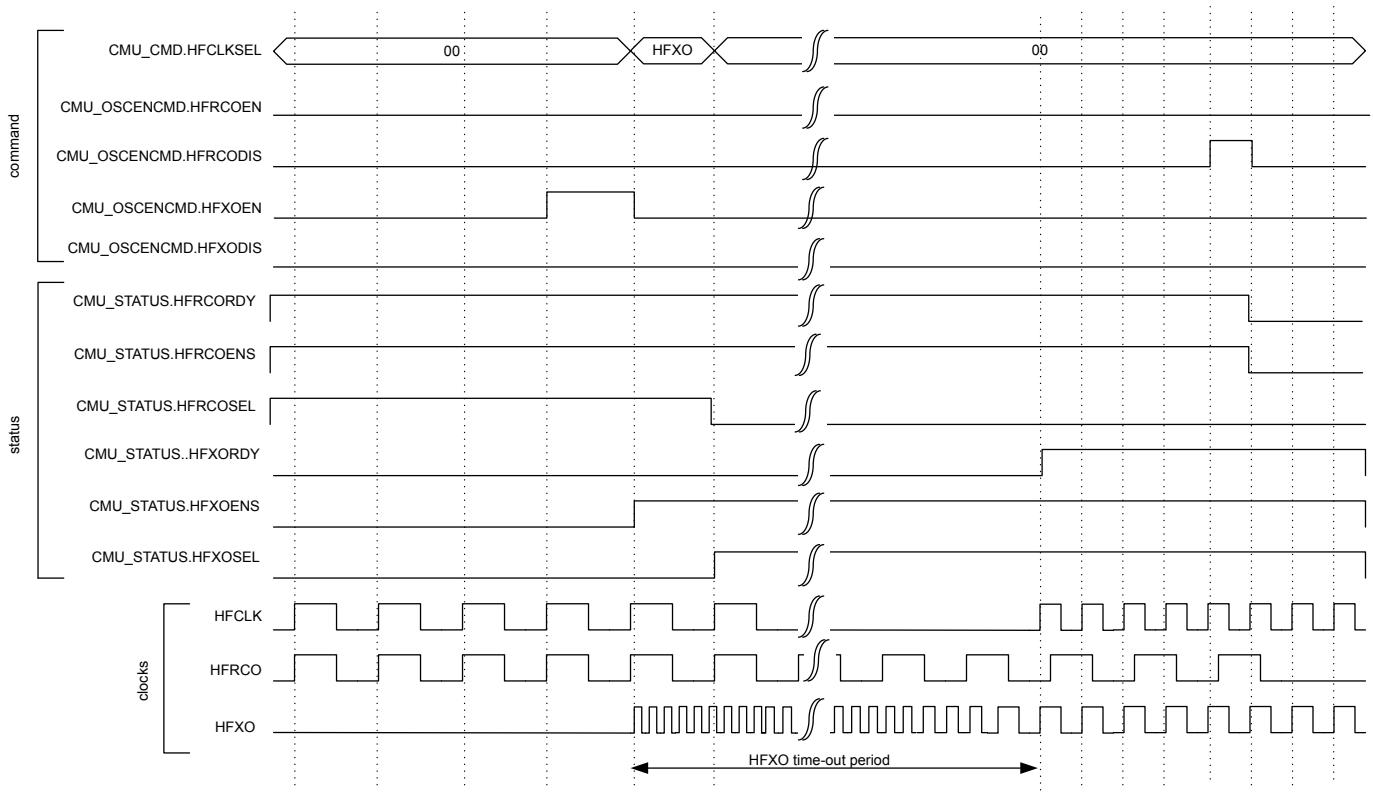


Figure 10.4. CMU Switching from HFRCO to HFXO before HFXO is ready

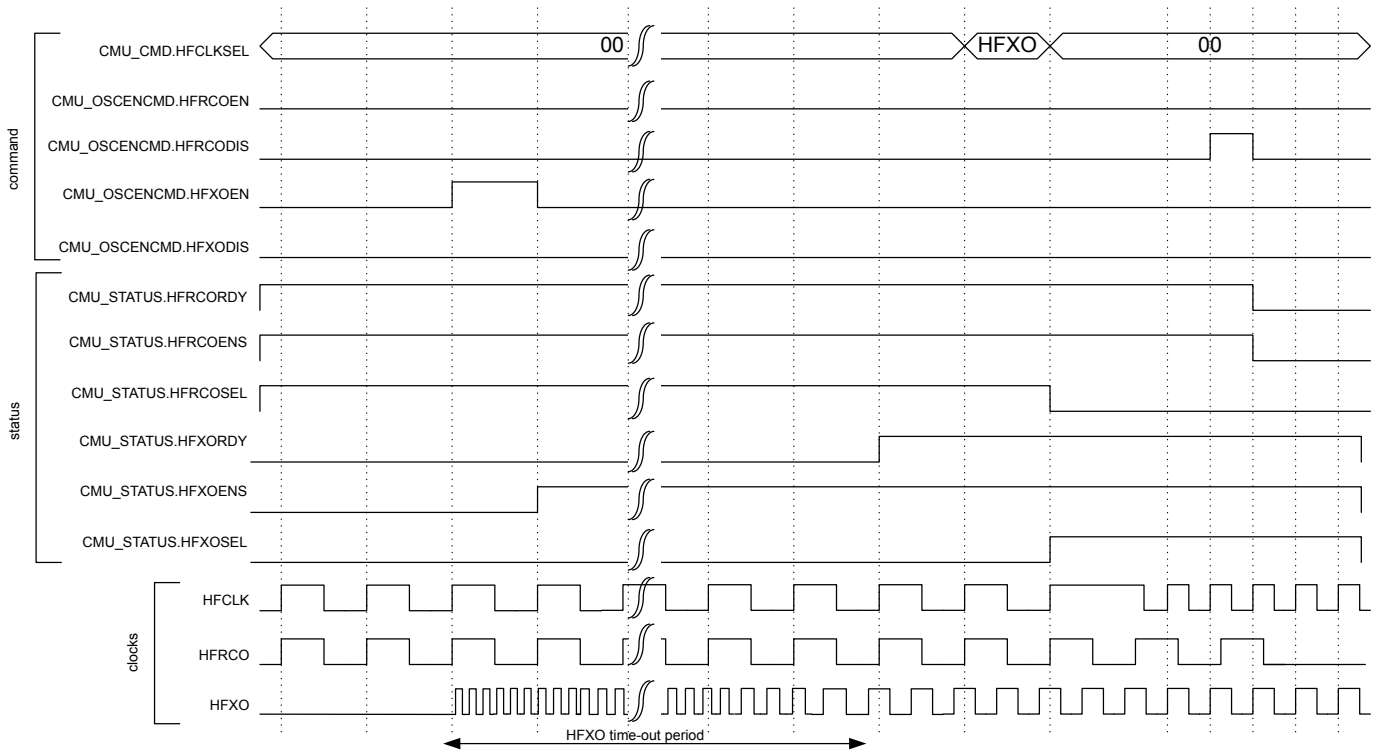


Figure 10.5. CMU Switching from HFRCO to HFXO after HFXO is ready

Switching clock source for LFACLK, LFBCLK,, LFCCLK and LFECLK is done by setting the LFA, LFB, LFC and LFE bitfields in CMU_LFACLKSEL, CMU_LFBCLKSEL, CMU_LFCCLKSEL and CMU_LFECLKSEL respectively. To ensure no stalls in the Low Energy Peripherals, the clock source should be ready before switching to it.

Note: To save energy, remember to turn off all oscillators not in use.

10.3.2.4 HFXO Configuration

The High Frequency Crystal Oscillator needs to be configured to ensure safe startup 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_N/HFXTAL_P pins as shown in Figure 10.6 HFXO Pin Connection on page 376

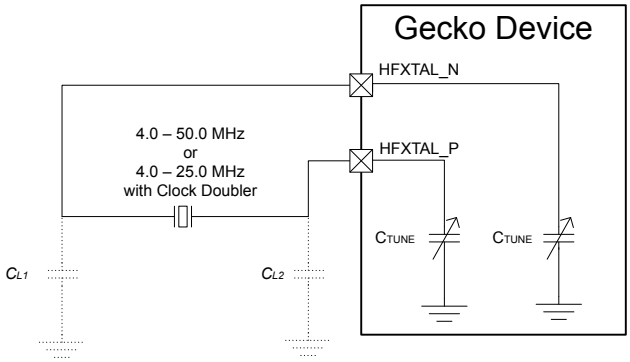


Figure 10.6. HFXO Pin Connection

By default the HFXO is started in crystal mode (XTAL), but it is possible to connect an active external sine wave or square wave clock source to the HFXTAL_N pin of the HFXO. By configuring the MODE field in CMU_HFXOCTRL to ACBUFEXTCLK (for external AC coupled sine wave) or DCBUFEXTCLK (for external DC coupled sine wave) or DIGEXTCLK (for external square wave), the HFXO can be bypassed and the source clock can be provided through the HFXTAL_N pin. When using external clock source, the HFXTAL_P pin is available to be used as regular GPIO.

An internal clock doubler can be used in crystal mode (XTAL) to generate a clock with double frequency compared to HFXO clock. By default the HFXOX2EN bitfield in CMU_HFXOCTRL is set, which makes this clock with double frequency available on HFXOX2. The use of double frequency on HFXOX2 is only allowed for crystals up to 25 MHz. If the frequency of the crystal is above 25 MHz, the HFXOX2EN must be set to 0 before enabling the HFXO. The clock doubler must also be disabled during 9.3.9 Voltage Scaling. If HFXOX2EN=0, the HFXO clock doubler is bypassed and therefore the HFXOX2 frequency will be equal to the HFXO frequency. When an external clock source is used, the clock doubler bypassed and the HFXOX2 frequency is always equal to the HFXO frequency.

Upon enabling the HFXO, a hardware state machine sequentially applies the configurable startup state and steady state control settings from the CMU_HFXOSTARTUPCTRL and CMU_HFXOSTEADYSTATECTRL registers. Configuration is required for both the startup state and the steady state of the HFXO. After reaching the steady operation state of the HFXO, further optimization can optionally be performed to optimize the HFXO for current consumption by an automatic Peak Detection Algorithm (PDA). HFXO operation is possible without PDA at the cost of higher current consumption than required. Furthermore, the oscillator amplitude can be kept stable by an automatic Peak Monitoring Algorithm (PMA) implemented in hardware. PMA is performed at every rising edge of ULFRCO, when it is enabled via the PEAKMONEN bitfield of CMU_HFXOSTEADYSTATECTRL register (enabled by default). PDA and PMA can only be activated in XTAL mode.

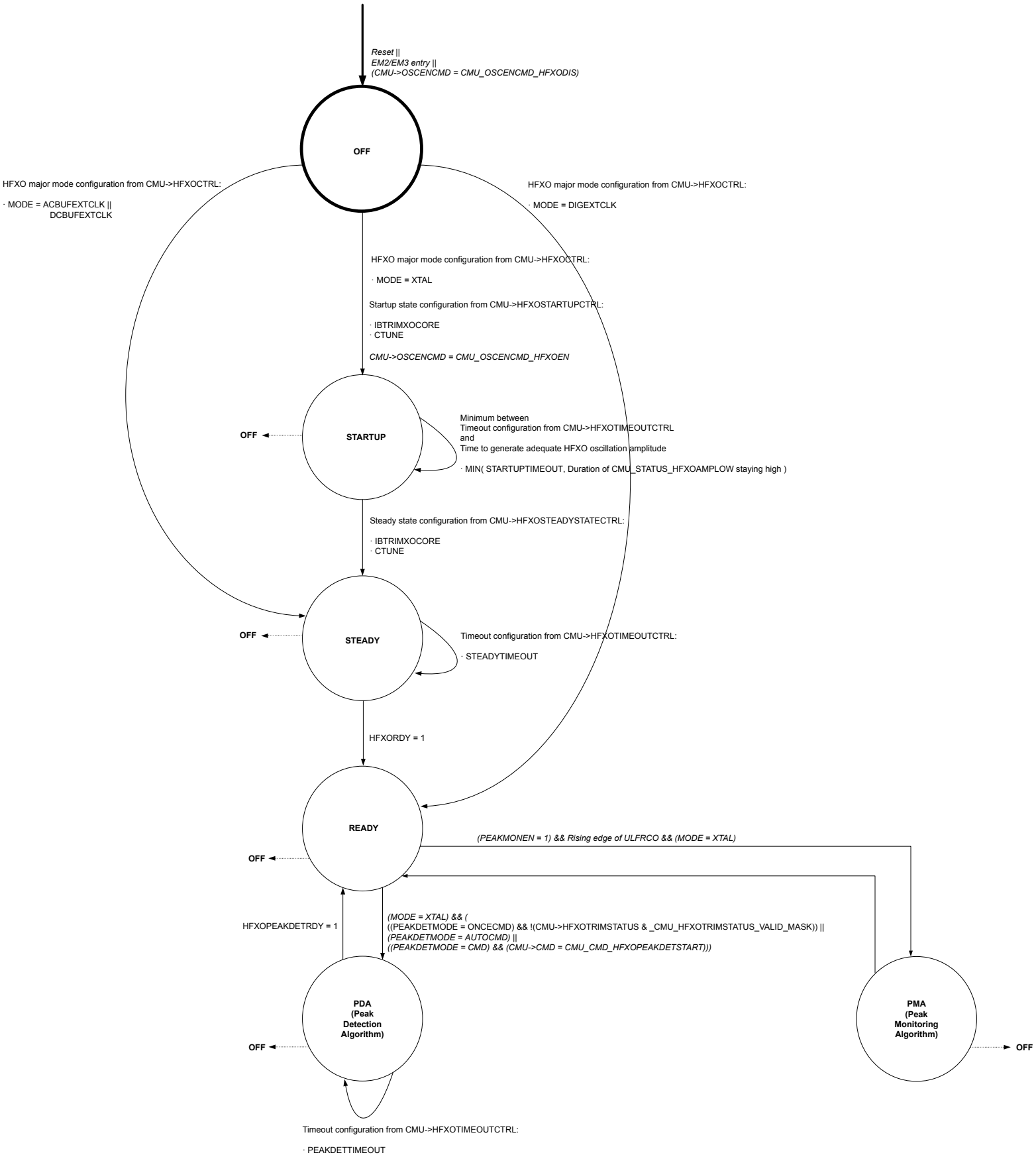


Figure 10.7. CMU HFXO control state machine

Refer to the device data sheet to find the configuration values for a given crystal. The startup state configuration needs to be written into the IBTRIMXOCORE and CTUNE bitfields of the CMU_HFXOSTARTUPCTRL register. The duration of the startup phase is configured in the STARTUPTIMEOUT bitfield of the CMU_HFXOTIMEOUTCTRL register. Similarly, the device data sheet provides the steady state configuration depending on the crystal's CL, RESR and oscillation frequency. This configuration is programmed into the IBTRIMXOCORE and CTUNE bitfields of the CMU_HFXOSTEADYSTATECTRL register. The duration of the steady phase is configured in the STEADYTIMEOUT bitfield of the CMU_HFXOTIMEOUTCTRL register.

All HFXO configuration needs to be performed prior to enabling the HFXO via HFXOEN in CMU_OSCENCMD unless noted otherwise. The HFXOENS flag in CMU_STATUS indicates if the HFXO has been successfully enabled. Once the HFXO startup time (STARTUPTIMEOUT plus STEADYTIMEOUT) has exceeded, the HFXO is ready for use as indicated by the HFXORDY flag in CMU_STATUS. If PDA is enabled, the HFXOPEAKDETRDY flag in the CMU_STATUS register indicates when this algorithm is ready and it is advised to also wait for this flag before using the HFXO.

The HFXO crystal bias current may be optimized and set to a value which decreases output phase noise without sacrificing PSR. This is done by programming the recommended IBTRIMXOCORE value into the CMU_HFXOSTEADYSTATECTRL register. The built-in Peak Detector Algorithm (PDA) performs further optimization to accommodate for process variations. Once PDA is ready as indicated by the HFXOPEAKDETRDY flag, the VALID flag in CMU_HFXOTRIMSTATUS register becomes 1 indicating that PDA found optimal bias current setting and this setting is available in the IBTRIMXOCORE bitfield of the CMU_HFXOTRIMSTATUS register. This IBTRIMXOCORE setting should be saved and can be applied directly during a future HFXO startup as a low power setting by programming it into the corresponding bitfield in CMU_HFXOSTEADYSTATECTRL while the HFXO is off. This is done automatically if HFXO is started with PEAKDETMODE register field of CMU_HFXOCTRL set to ONCECMD and in this case PDA is skipped upon repeated HFXO startup.

Default PDA is started automatically once the HFXO has become ready. Repeated PDA can be triggered by writing HFXOPEAKDETSTART to 1 in the CMU_CMD register. PDA can also be triggered only by the command register by configuring PEAKDETMODE to CMD in the CMU_HFXOCTRL register before starting the HFXO. The PEAKDETTIMEOUT bitfield in the CMU_HFXOTIMEOUTCTRL register is used to time the PDA steps and needs to be configured according to the device data sheet for the given crystal. The PEAKDETEN bitfield of the CMU_HFXOSTEADYSTATECTRL register is only used during manual (i.e. fully software controlled) peak detection and is ignored during automatic or command based triggering of the PDA. Note that the manual PDA mode is not recommended for general usage and therefore it is not further described. PDA and PMA should not be used when using an external wave as clock source.

10.3.2.4.1 Automatic HFXO Start

The enabling of the HFXO and its selection as HFSRCCLK source can be performed automatically by hardware. Automatic control of the HFXO is controlled via the AUTOSTARTSELEM0EM1 and AUTOSTARTEM0EM1 bits in the CMU_HFXOCTRL register. It further depends on the energy mode of the EFM32 .

An automatic HFXO enable is performed only if any of the following conditions are met:

- EFM32 is in EM0/EM1 and AUTOSTARTEM0EM1 or AUTOSTARTSELEM0EM1 are set to 1.

An automatic HFXO select is performed only if any of the following conditions is met:

- EFM32 is in EM0/EM1 and AUTOSTARTSELEM0EM1 is set to 1.

Whenever any of the conditions for automatic HFXO enable is met, software is not allowed to disable the HFXO. An attempt to do so (e.g., by writing 1 to the HFXODIS bit) is ignored and causes the HFXODISERR bit in the CMU_IF register to be set to 1. Similarly, whenever any of the conditions for automatic HFXO selection is met, software is not allowed to deselect the HFXO as clock source for HFSRCCLK. An attempt to do so (e.g., by selecting another clock source via CMU_HFCLKSEL) is ignored and causes the HFXODISERR bit in the CMU_IF register to be set to 1. Note that CMUERR is not implied by HFXODISERR. CMUERR will not get set to 1 for the above scenarios in which HFXODISERR gets set.

Software can only disable or deselect the HFXO after removing all of the HFXO automatic enable or select reasons. The HFXO is only disabled by hardware upon EM2, EM3 or EM4 entry.

In case that AUTOSTARTSELEM0EM1 is set to 1 in EM0/EM1 (irrespective of the other autostart bits), the HFXO select will occur immediately, even if HFXO is not ready yet. Upon wake-up into EM0/EM1 this can therefore lead to a relatively long startup time as the system will not start operating from the HFRCO as it would otherwise do.

Note that the user should take care that the settings in the MSC_READCTRL and CMU_CTRL registers, as described in 10.3.3 Configuration for Operating Frequencies, are compatible with HFXO frequency before enabling the HFXO automatic startup feature. A basic automatic HFXO start scenario is shown in Figure 10.8 CMU Automatic Startup and Selection of HFXO on page 379.

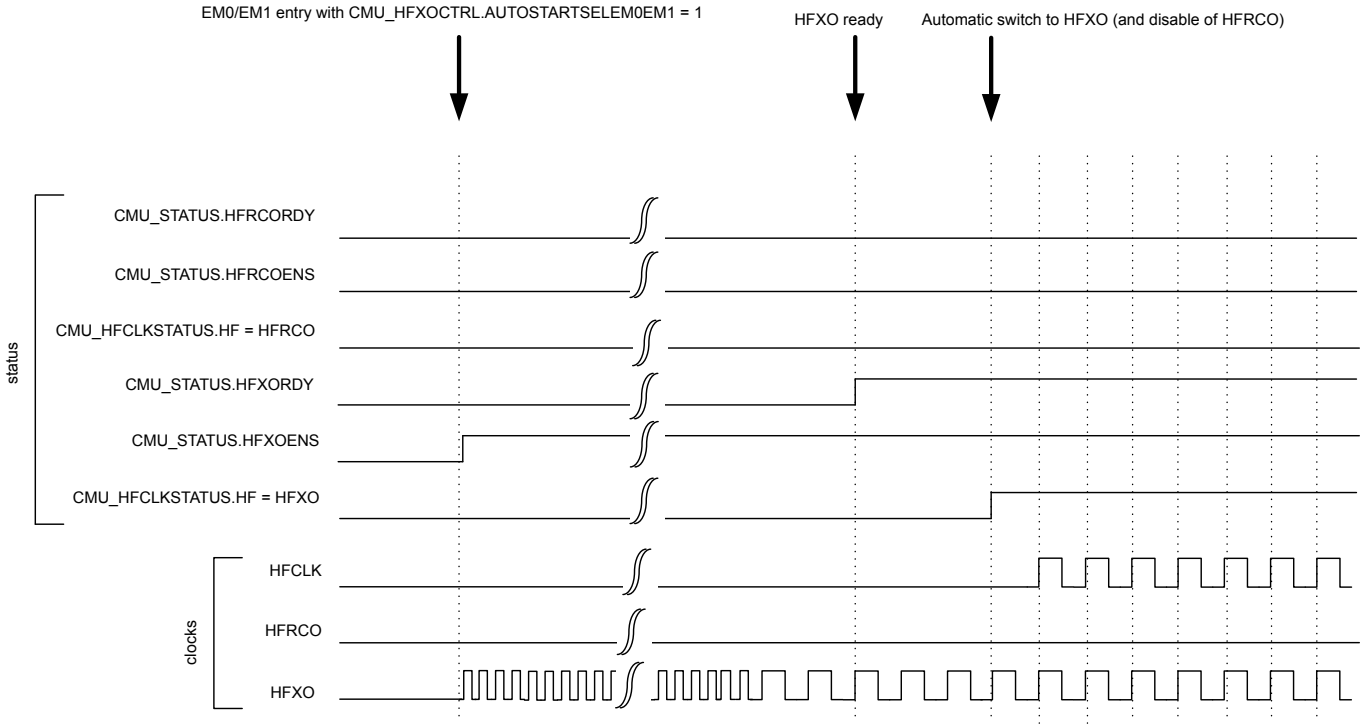


Figure 10.8. CMU Automatic Startup and Selection of HFXO

If an automatic selection of HFXO is performed, which switches the clock source used for HFSRCCLK, then the HFXOAUTOSW bit in CMU_IF is set to 1. After automatic enable and selection of the HFXO, the HFRCO is automatically disabled in case it is running. The disabling of a running HFRCO is signalled via the HFRCODIS bit in CMU_IF. This only applies to the HFRCO. If for example the LFXO was used as HFSRCCLK at the time of automatic selection of the HFXO, the LFXO remains unaffected.

The interaction between automatic HFXO startup and selection with startup and selection of HFRCO is shown in Figure 10.9 CMU HFRCO Startup/Selection While Awaiting Automatic HFXO Startup/Selection on page 380.

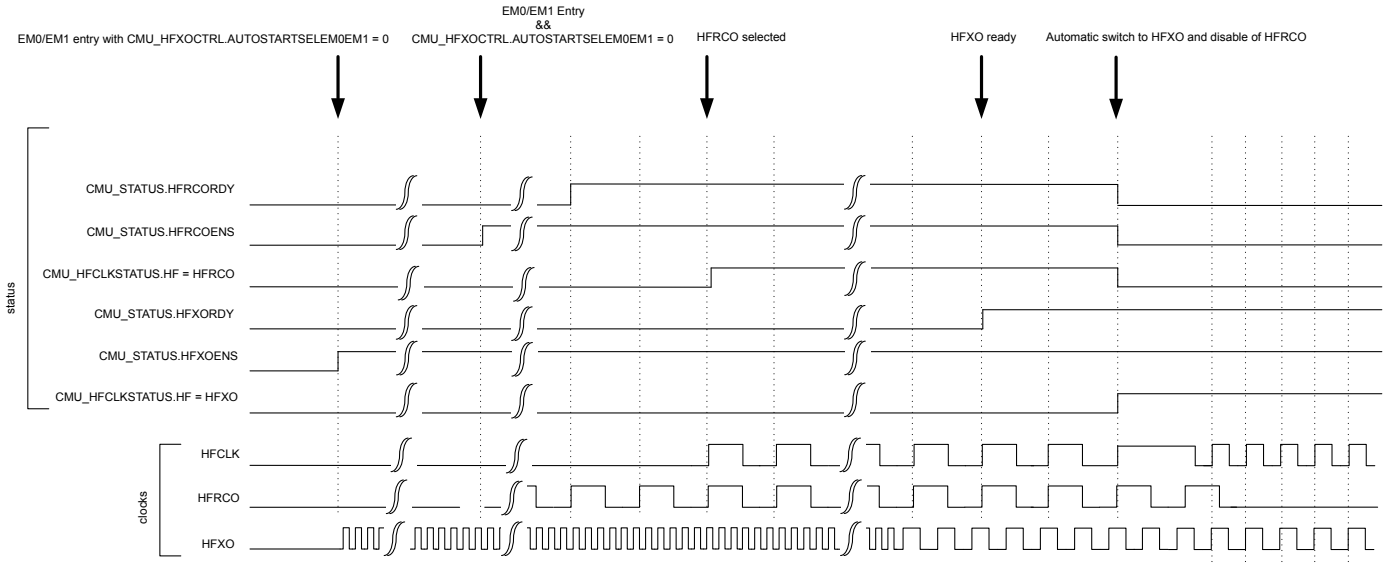


Figure 10.9. CMU HFRCO Startup/Selection While Awaiting Automatic HFXO Startup/Selection

10.3.2.5 LFXO Configuration

The Low Frequency Crystal Oscillator (LFXO) is default configured to ensure safe startup for all crystals. In order to optimize startup time and power consumption for a given crystal, it is possible to adjust the startup gain in the oscillator by programming the GAIN field in CMU_LFXOCTRL. Recommendations for the GAIN setting are as follows:

1. C0 must be < 2 pF
2. For 12.5 pF < CL < 18 pF, GAIN = 3
3. For 8 pF < CL < 12.5 pF, GAIN = 2
4. For 6 pF < CL < 8 pF, GAIN = 1
5. For CL = 6 pF, GAIN = 0

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 LFXO can be retained on in EM4 Hibernate/Shutoff. In that case its required configuration is latched/retained throughout EM4 even though the CMU_LFXOCTRL register itself will be reset. Upon EM4 exit, the CMU_LFXOCTRL register therefore needs to be reconfigured to its original settings and the LFXO needs to be restarted via CMU_OSCENCMD, before optionally unlatching the retained LFXO configuration by writing 1 to EM4UNLATCH in the EMU_CMD register. The LFXO startup time is configured via the TIMEOUT bitfield of the CMU_LFXOCTRL register. If the LFXO has been retained throughout EM4 Hibernate/Shutoff, it can be quickly restarted for use as LFACLK, LFBCLK, LFCCLK or LFECLK by using its minimum TIMEOUT setting. While retained, the LFXO can be used down to EM4 Hibernate as source for LFECLK and down to EM4 Shutoff as source for CRYOCLK.

The LFXO crystal is connected to the LFXTAL_N/LFXTAL_P pins as shown in Figure 10.10 LFXO Pin Connection on page 381.

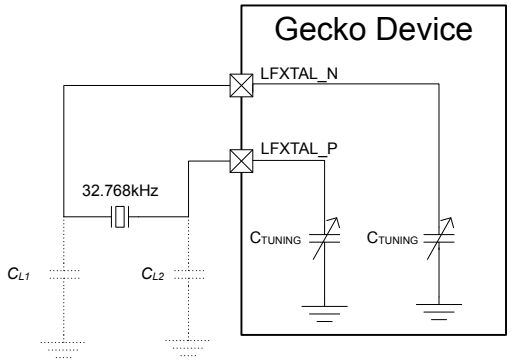


Figure 10.10. LFXO Pin Connection

By configuring the MODE field in CMU_LFXOCTRL, the LFXO can be bypassed, and an external clock source can be connected to the LFXTAL_N pin of the LFXO oscillator. If MODE is set to BUFEXTCLK, an external active sine source can be used as clock source. If MODE is set to DIGEXTCLK, an external active CMOS source can be used as clock source.

The LFXO includes on-chip tunable capacitance, which can replace external load capacitors. The TUNING bitfield of the CMU_LFXOCTRL register is used to tune the internal load capacitance connected between LFXTAL_P and ground and LFXTAL_N and ground symmetrically. The capacitance range and step size information is available in the device data sheets. Use the formula below to calculate the TUNING bitfield:

$$TUNING = ((desiredTotalLoadCap * 2 - Min(C_{LFXO\_T})) / C_{LFXO\_TS})$$

Figure 10.11. CMU LFXO Tuning Capacitance Equation

These tunable capacitors can also be used to compensate for temperature drift of the XTAL in software. Crystals normally have a temperature dependency which is given by a parabolic function. The crystal has highest frequency at its turnover temperature, normally 25C. The frequency is reduced following a parabola for higher and lower temperatures. The LFXO offers a mechanism to internally add capacitance on the LFXTAL_N and LFXTAL_P pins (in parallel to an optional external load capacitance). The variation in frequency as a function of temperature can therefore be compensated by adjusting the load capacitance. When the temperature compensation scheme is used, the maximum internal capacitance should be used to obtain good frequency matching at the turnover temperature. For higher and lower temperatures software then has the maximum range available to adjust the tuning. The external load capacitance

must then of course be reduced accordingly. Note that the ADC0 (28. ADC - Analog to Digital Converter) includes an embedded temperature sensor and that the EMU (9. EMU - Energy Management Unit) offers a temperature management interface, both of which can be used in combination with this LFXO temperature compensation scheme.

The XTAL oscillation amplitude can be controlled via the HIGHAMPL bitfield in CMU_LFXOCTRL. Setting HIGHAMPL to 1 will result in higher amplitude, which in turn provides safer operation, somewhat improved duty cycle, and lower sensitivity to noise at the cost of increased current consumption.

The AGC bit of the CMU_LFXOCTRL register is used to turn on or off the Automatic Gain Control module that adjusts the amplitude of the XTAL. When disabled, the LFXO will run at the startup current and the XTAL will oscillate rail to rail, again providing safer operation, improved duty cycle, and lower sensitivity to noise at the cost of increased current consumption.

10.3.2.6 HFRCO, USHFRCO and AUXHFRCO Configuration

It is possible to calibrate the HFRCO, USHFRCO and AUXHFRCO to achieve higher accuracy (see the device data sheets for details on accuracy). The frequency is adjusted by changing the TUNING and FINETUNING bitfields in CMU_HFRCOCTRL, CMU_USHFRCOCTRL and CMU_AUXHFRCOCTRL. Changing to a higher value will result in a lower frequency. Refer to the data sheet for stepsize details.

The HFRCO can be set to one of several different frequency bands from 1 MHz to 72 MHz by setting the FREQRANGE field in CMU_HFRCOCTRL. Similarly the AUXHFRCO can be set to one of several different frequency bands from 1 MHz to 50 MHz by setting the FREQRANGE field in CMU_AUXHFRCOCTRL. The USHFRCO can be set to one of several different frequency bands from 1 MHz to 50 MHz by setting the FREQRANGE field in CMU_USHFRCOCTRL. The HFRCO, USHFRCO and AUXHFRCO frequency bands are calibrated during production test, and the production tested calibration values can be read from the Device Information (DI) page. The DI page contains separate tuning values for various frequency bands. During reset, HFRCO and AUXHFRCO tuning values are set to the production calibrated values for the 19 MHz band, which is the default frequency band. The USHFRCO default band is different and is set to 48 MHz. When changing to a different HFRCO, USHFRCO or AUXHFRCO band, make sure to also update the TUNING value and other bitfields in the CMU_HFRCOCTRL, CMU_USHFRCOCTRL and CMU_AUXHFRCOCTRL registers. Typically the entire register is written with a value obtained from the Device Information (DI) page. Refer to 4.6 DI Page Entry Map for information on which frequency band settings are stored in the DI page.

The frequency can be tuned more accurately via the FINETUNING bitfield if fine tuning has been enabled via the FINETUNINGEN bit. Note that there will be a slight increase in the oscillator current consumption when fine tuning is enabled. Note also that changing the value of FINETUNINGEN will result in a frequency shift, regardless of the FINETUNING field value. If the oscillator is to be used at different times with fine tuning enabled and disabled, it should be tuned separately for both settings. The HFRCO, USHFRCO and AUXHFRCO contain a local prescaler, which can be used in combination with any FREQRANGE setting. These prescalers allow the output clocks to be divided by 1, 2, or 4 as configured in the CLKDIV bitfield.

When using 10.3.2.8 RC Oscillator Calibration to tune HFRCO, USHFRCO and AUXHFRCO to the desired frequency, linear search must be used to avoid over clocking the calibration counters. Before changing the FREQRANGE field in CMU_HFRCOCTRL, TUNING and FINETUNING fields should initially be set to the highest value (slowest frequency). After changing the FREQRANGE, linearly step TUNING value until desired frequency is reached. Likewise, before changing the TUNING field, FINETUNING field should initially be set to the highest value (lowest frequency). After changing the TUNING field, linearly step FINETUNING until accuracy is reached.

10.3.2.7 LFRCO Configuration

It is possible to calibrate the LFRCO to achieve higher accuracy (see the device data sheets for details on accuracy). The frequency is adjusted by changing the TUNING bitfield in CMU_LFRCOCTRL. Changing to a higher value will result in a lower frequency. Refer to the data sheet for stepsize details.

The LFRCO can be retained on in EM4 Hibernate/Shutoff. In that case its required configuration is latched/retained throughout EM4 even though the CMU_LFRCOCTRL register itself will be reset. Upon EM4 exit the CMU_LFRCOCTRL register therefore needs to be reconfigured to its original settings and the LFRCO needs to be restarted via CMU_OSCENCMD, before optionally unlatching the retained LFRCO configuration by writing 1 to EM4UNLATCH in the EMU_CMD register. The LFRCO startup time is configured via the TIMEOUT bitfield of the CMU_LFRCOCTRL register. Default its 16 cycle startup should be used. However, in case the LFRCO has been retained throughout EM4 Hibernate/Shutoff, it can be quickly restarted for use as LFACLK, LFBCLK or LFCCLK by using its minimum TIMEOUT setting. While retained, the LFRCO can be used down to EM4 Hibernate as source for LFECLK and down to EM4 Shutoff as source for CRYOCLK.

The LFRCO is also calibrated in production and its TUNING values are set to the correct value during reset.

The LFRCO can be put in duty cycle mode by setting the ENVREF bit in CMU_LFRCOCTRL to 1 before starting the LFRCO. This will reduce current consumption, but will result in slightly worse accuracy especially at high temperatures. Setting the ENCHOP and/or ENDEM bitfields to 1 in the CMU_LFRCOCTRL register will improve the average LFRCO frequency accuracy at the cost of a worse cycle-to-cycle accuracy.

10.3.2.8 RC Oscillator Calibration

The CMU has built-in HW support to efficiently calibrate the RC oscillators (LFRCO, HFRCO, USHFRCO, AUXHFRCO, etc) at run-time. For a complete list of supported oscillators, refer to DOWNSEL and UPSEL fields in CMU_CALCTRL. See Figure 10.12 HW-support for RC Oscillator Calibration on page 383 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. The top value for the down-counter must be written to CMU_CALCNT before calibration is started. The down-counter counts for CMU_CALCNT+1 cycles. When the down-counter has reached 0, the up-counter is sampled and the CALRDY interrupt flag 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 and 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 top+1 and sample+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 stopped by writing CALSTOP in CMU_CMD. With this HW support, it is simple to write efficient calibration algorithms in software.

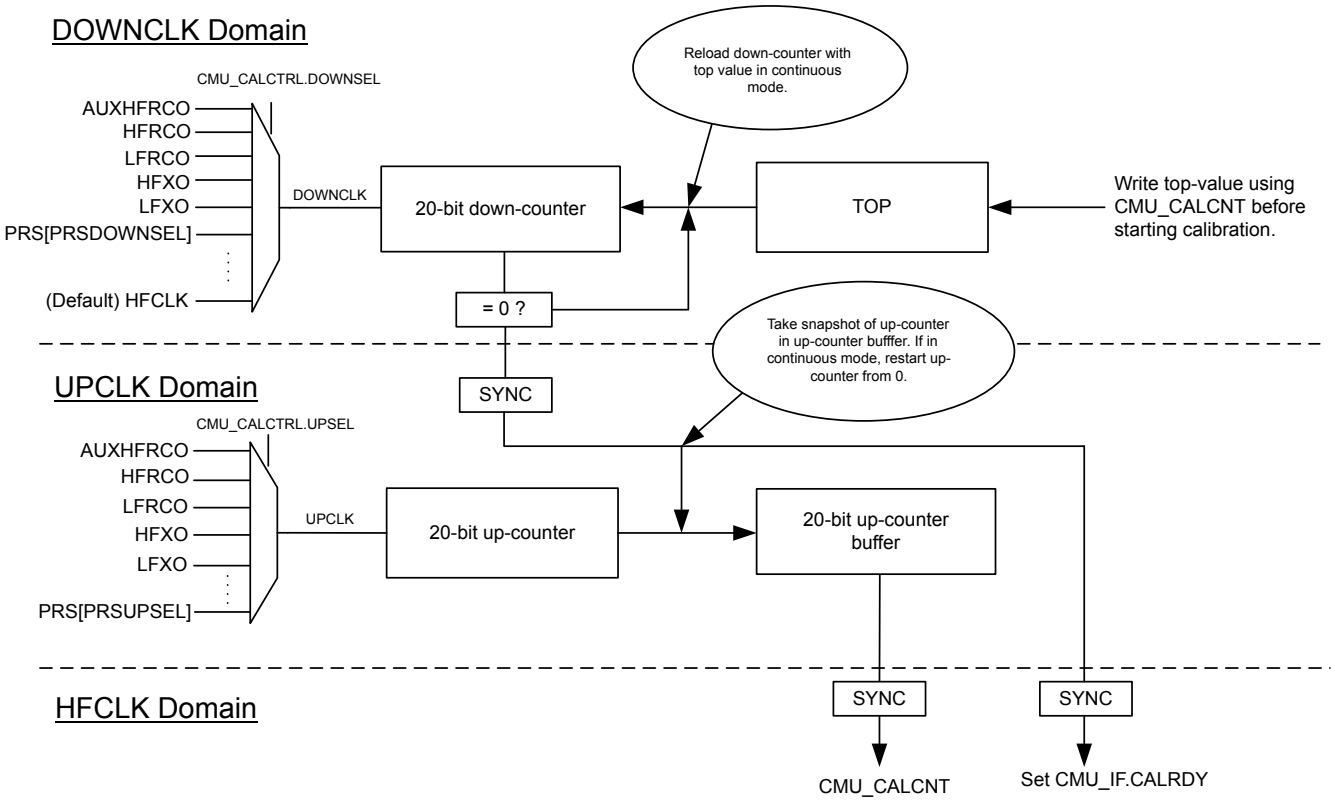


Figure 10.12. HW-support for RC Oscillator Calibration

The counter operation for single and continuous mode are shown in Figure 10.13 Single Calibration (CONT=0) on page 384 and Figure 10.14 Continuous Calibration (CONT=1) on page 384 respectively.

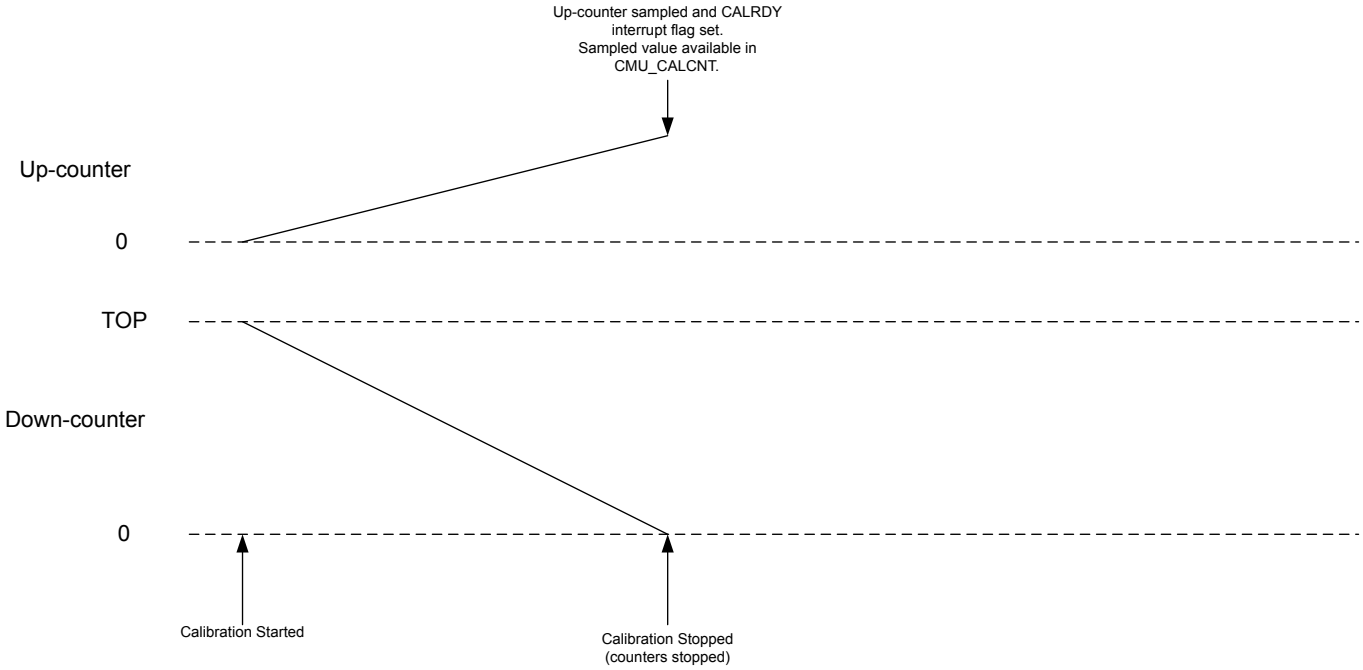


Figure 10.13. Single Calibration (CONT=0)

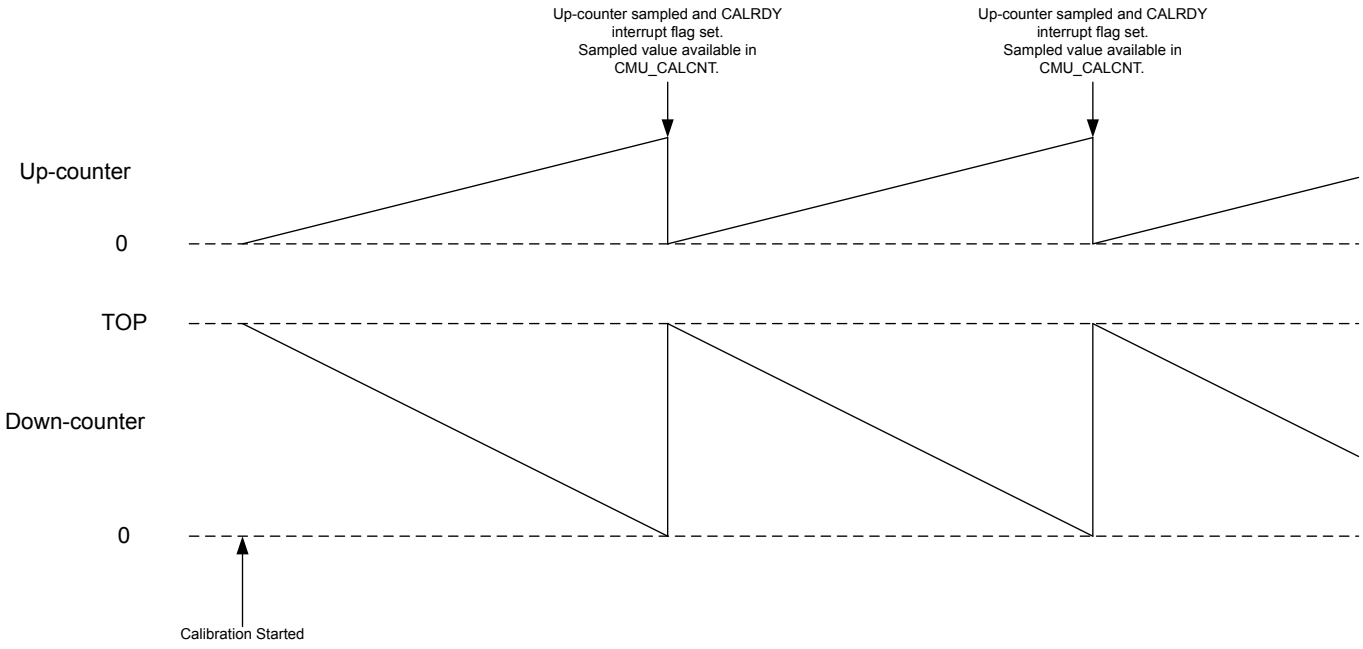


Figure 10.14. Continuous Calibration (CONT=1)

10.3.3 Configuration for Operating Frequencies

The HFXO is capable of frequencies up to 50 MHz, which allows the EFM32 to run at up to this frequency. However, not all High Frequency clocks are allowed to run at this maximum frequency. Clocks need to be limited to the frequencies shown in Table 10.4 Maximum Allowed Clock Frequencies on page 385, for example by prescaling them or by selecting an appropriate clock source. Also modules such as the Memory System Controller (MSC), Low Energy Peripheral Interface, and DMEM must be configured correctly to allow operation at higher frequencies as explained further below.

Table 10.4. Maximum Allowed Clock Frequencies

Clock	VSCALE2 (1.2V)	VSCALE0 (1.0V)
HFRCO	<= 72 MHz	<= 20 MHz
AUXHFRCO, USHFRCO	<= 50 MHz	<= 20 MHz
HFXO	<= 50 MHz	<= 20 MHz[1]
HFXOX2	<= 50 MHz	Off[2]
HFSRCCLK, HFPERBCLK	<= 72 MHz	<= 20 MHz
HFBUSCLK, HFPERCLK, HFPERCCLK	<= 50 MHz	<= 20 MHz
QSPInCLK	<= 50 MHz	<= 20 MHz
SDIOCLK	<= 50 MHz	Off[3]
USBCLK	48 MHz	32 kHz or off
$HFCLK_{SDIO}$	<= 72 MHz	Off[3]
DBGCLK	<= 36 MHz	<= 10 MHz

Note:
1. 50 MHz allowed when not selected.
2. HFXO clock doubling is not supported when using voltage scaling.
3. User should turn clock off when using voltage scaling.

The MODE bitfield in MSC_READCTRL makes sure the flash is able to operate at the given HFCLK frequency by inserting wait states for flash accesses. The required settings for controlling flash wait states are shown in Table 10.5 MSC Configuration for Operating Frequencies, at VSCALE2: Flash Wait States on page 385. The WSHFLE bitfield in CMU_CTRL is used to ensure that the Low Energy Peripheral Interface is able to operate at the given $HFCLK_{LE}$ frequency by inserting wait states when using this interface. The required settings are shown in Table 10.10 LE Configuration for Operating Frequencies: Low Energy Peripheral Interface on page 386.

The RAMxWSEN and RAMxPREFETCHEN bits in MSC_RAMCTRL serve to add wait states to RAM accesses when necessary. Refer to Table 10.8 DMEM Configuration for Operating Frequencies, at VSCALE2: RAM Wait State Enable on page 386 and Table 10.9 DMEM Configuration for Operating Frequencies, at VSCALE0: RAM Wait State Enable on page 386. When RAMxWSEN is set to 1, any read from an address in the corresponding RAM bank may incur one extra wait state cycle. RAMxPREFETCHEN operates in a similar way, except that sequential burst reads will only incur the extra wait state cycle at the beginning of the burst. Using RAMxPREFETCHEN is more efficient for sequential burst reads, with a slight increase in power consumption.

Before going to a high frequency, make sure the registers in the table have the correct values. When going down in frequency, make sure to keep the registers at the values required by the higher frequency until after the switch has been done.

Table 10.5. MSC Configuration for Operating Frequencies, at VSCALE2: Flash Wait States

Condition	MODE in MSC_READCTRL
HFCLK <= 18 MHz	WS0 or above
18 MHz < HFCLK <= 36 MHz	WS1 or above
36 MHz < HFCLK <= 54 MHz	WS2 or above
HFCLK > 54 MHz	WS3

Table 10.6. MSC Configuration for Operating Frequencies, at VSCALE0: Flash Wait States

Condition	MODE in MSC_READCTRL
HFCLK <= 7 MHz	WS0 or above
7 MHz < HFCLK <= 14 MHz	WS1 or above
14 MHz < HFCLK <= 20 MHz	WS2

Table 10.7. Bus System Configuration for Operating Frequencies, at VSCALE2. (at WS0 is sufficient): Peripheral Access Wait Modes

Condition	WAITMODE in MSC_CTRL
HFCLK <= 50 MHz	WS0 or above
50 MHz < HFCLK <= 72 MHz	WS1

Table 10.8. DMEM Configuration for Operating Frequencies, at VSCALE2: RAM Wait State Enable

Condition	RAMxWSEN or RAMxPREFETCHEN in MSC_RAMCTRL
HFCLK <= 38 MHz	0
38 MHz < HFCLK <= 72 MHz	1

Table 10.9. DMEM Configuration for Operating Frequencies, at VSCALE0: RAM Wait State Enable

Condition	RAMxWSEN or RAMxPREFETCHEN in MSC_RAMCTRL
HFCLK <= 16 MHz	0
16 MHz < HFCLK <= 20 MHz	1

Table 10.10. LE Configuration for Operating Frequencies: Low Energy Peripheral Interface

Condition	WSHFLE in CMU_CTRL
$HFCLK_{LE}$ <= 32 MHz	0 / 1
$HFCLK_{LE}$ > 32 MHz	1

10.3.4 Energy Modes

The availability of oscillators and system clocks depends on the chosen energy mode. Default the high frequency oscillators (HFRCO, USHFRCO, AUXHFRCO, and HFXO) and high frequency clocks (HFSRCLK, HFCLK, HFCORECLK, HFBUSCLK, HFPERCLK, HFPERBCLK, HFPERCCLK, HFCLKLE) are available downto EM1 Sleep. From EM2 DeepSleep onwards these oscillators and clocks are normally off, although special cases exist as summarized in Table 10.11 Oscillator and Clock Availability in Energy Modes on page 387 and Table 9.2 EMU Energy Mode Overview on page 270. The CMU overview figure in Figure 10.1 CMU Overview - High Frequency Portion on page 362 and Figure 10.2 CMU Overview - Low Frequency Portion on page 363 also indicate 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. Default these oscillators are also off in EM4 Hibernate and EM4 Shutoff, but they can be retained on in these states as well if needed. The ultra low frequency oscillator (ULFRCO) is default on in all energy modes, except for EM4 Shutoff, but it can be retained on in that mode as well if needed. The low frequency clocks (LFACLK, LFBCLK, LFCCLK, LFECLK, WDOGnCLK, and CRYOCLK) 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 10.11 Oscillator and Clock Availability in Energy Modes on page 387.

Table 10.11. Oscillator and Clock Availability in Energy Modes

	EM0 Active/EM1 Sleep	EM2 DeepSleep	EM3 Stop	EM4 Hibernate	EM4 Shutoff
HFRCO	On[1]	Off	Off	Off	Off
HFXO	On[1]	Off	Off	Off	Off
USHFRCO	On[1]	Off	Off	Off	Off
AUXHFRCO	On[1]	On[2]	On[2]	Off	Off
LFRCO, LFXO	On[1]	On[1]	Off	Retained on[3]	Retained on[3]
ULFRCO	On	On	On	On	Retained on[3]
HFSRCLK, HFCLK, HFCORECLK, HFBUSCLK, HFPERCLK, HFPERBCLK, HFPERCCLK, HFCLKLE	On[1]	Off	Off	Off	Off
QSPInCLK	On[1]	Off[4]	Off[4]	Off	Off
SDIOCLK	On[1]	Off[4]	Off[4]	Off	Off
PDMCLK	On[1]	Off[4]	Off[4]	Off	Off
USBCLK	On[1]	On[5]	Off	Off	Off
AUXCLK	On[1]	On[2]	On[2]	Off	Off
ADCnCLK	On[1]	On[6]	On[6]	Off	Off
LFACLK, LFBCLK, LFCCLK	On[1]	On[1]	On[7]	Off	Off
LFECLK	On[1]	On[1]	On[7]	Retained on[3]	Off
WDOGnCLK	On[1]	On[1]	On[7]	Off	Off
CRYOCLK	On[1]	On[1]	On[7]	Retained on[3]	Retained on[3]

	EM0 Active/EM1 Sleep	EM2 DeepSleep	EM3 Stop	EM4 Hibernate	EM4 Shutoff

Note:

1. Under software control.
2. Default off, but kept active if used by the ADC.
3. Default off, but can be retained on.
4. Must be turned off by software before EM2/3 entry.
5. Will be on if enabled and running LFXO or LFRCO is selected.
6. Will be kept on if AUXHFRCO is selected as clock source.
7. On only if ULFRCO is used as clock source.

10.3.5 Clock Output on a Pin

It is possible to configure the CMU to output clocks on the CMU_CLK0, CMU_CLK1 and CMU_CLK2 pins. This clock selection is done using the CLKOUTSEL0, CLKOUTSEL1 and CLKOUTSEL2 bitfields respectively in CMU_CTRL. The required output pins must be enabled in the CMU_ROUTEPEN register and the pin locations can be configured in the CMU_ROUTELOC0 register. The following clocks can be output on a pin:

- HFSRCCLK and HFEXPCLK. The HFSRCCLK is the high frequency clock before any prescaling has been applied. The HFEXPCLK is a prescaled version of HFCLK as controlled by the HFEXPPRESC bitfield in the CMU_HFPRESC register.
- The unqualified clock output from any of the oscillators (ULFRCO, LFRCO, LFXO, HFXO). Note that these unqualified clocks can exhibit glitches or skewed duty-cycle during startup and therefore these clock outputs are normally not used before observing the related ready flag being set to 1 in CMU_STATUS.
- The qualified clock from any of the oscillators (ULFRCO, LFRCO, LFXO, HFXO, HFRCO, USHFRCO, AUXHFRCO). A qualified clock will not have any glitches or skewed duty-cycle during startup. For LFRCO, LFXO and HFXO correct configuration of the TIMEOUT bitfield(s) in CMU_LFRCOCTRL, CMU_LFXOCTRL and CMU_HFXOTIMEOUTCTRL respectively is required to guarantee a properly qualified clock.
- The qualified HFXO clock divided by 2 (HFXODIV2Q).

HFCLK will not have a 50-50 duty cycle when any other division factor than 1 is used in CMU_HFPRESC (i.e. if PRESC is not equal to 0). In such a case, the exported HFEXPCLK will therefore also not be 50-50 when its division factor is not set to an even number in CMU_HFEXPPRESC.

10.3.6 Clock Input From a Pin

It is possible to configure the CMU to input a low-frequency (< 1 MHz) clock from the CMU_CLKI0. This clock can be selected to drive HFSRCCLK and DPLL reference using CMU_HFCLKSEL and CMU_DPLLCTRL respectively. The required input pin must be enabled in the CMU_ROUTEPEN register and the pin location can be configured in the CMU_ROUTELOC1 register.

10.3.7 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 CMUCLKOUT0, CMUCLKOUT1 and CMUCLKOUT2. The clocks which can be produced via CMUCLKOUT0, CMUCLKOUT1 and CMUCLKOUT2 are selected via the CLKOUTSEL0, CLKOUTSEL1 and CLKOUTSEL2 fields respectively in CMU_CTRL.

Note that the CLKOUTSEL0 and CLKOUTSEL1 fields are also used for selecting which clock is output onto a pin as described in 10.3.5 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 CMU_ROUTEPEN and CMU_ROUTELOC0 registers.

10.3.8 Error Handling

Certain restrictions apply to how and when the CMU registers can be configured as is described for the respective registers. Not adhering to these restrictions can lead to unpredictable and non-defined behaviour. Some of these software restrictions are checked in hardware and not adhering to them will cause the CMUERR interrupt flag in CMU_IF to be set to 1. The restrictions impacting CMUERR are as follows:

- CMU_HFRCOCTRL should not be written while HFRCOBSY in the CMU_SYNCBUSY register is set to 1.
- CMU_USHFRCOCTRL should not be written while USHFRCOBSY in the CMU_SYNCBUSY register is set to 1.
- CMU_AUXHFRCOCTRL should not be written while AUXHFRCOBSY in the CMU_SYNCBUSY register is set to 1.
- CMU_HFXOSTARTUPCTRL, CMU_HFXOSTEADYSTATECTRL and CMU_HFXOTIMEOUTCTRL should not be written while HFXOBSY in the CMU_SYNCBUSY register is set to 1. Note that writes to CMU_HFXOCTRL do not impact CMUERR. Although most of its bitfields need to be configured before enabling the HFXO, it it allowed to change the AUTOSTART bits (i.e. AUTOSTARTSELEM0EM1 and AUTOSTARTEM0EM1) at any time.
- HFXO should not be enabled before it has been properly disabled (so only enable HFXO when HFXOENS=0 or HFXOBSY=0). Likewise, HFXO should not be disabled before it has been properly enabled (so only disable HFXO when HFXOENS=1 or HFXOBSY=0).
- CMU_LFRCOCTRL should not be written while LFRCOBSY in the CMU_SYNCBUSY register is set to 1. The GMCCURTUNE bitfield should not be written with a differing value while the LFRCOVREFBSY flag is set to 1.
- CMU_LFXOCTRL should not be written while LFXOBSY in the CMU_SYNCBUSY register is set to 1.

10.3.9 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.

10.3.10 Wake-up

The CMU can be (partially) active all the way down to EM4 Shutoff. It can wake up the CPU from EM2 upon LFRCO or LFXO becoming ready as LFRCORDY and LFXORDY can be used as wake-up interrupt.

10.3.11 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.

10.3.12 Digital Phase-Locked Loop

The Digital Phase-Locked Loop (DPLL) uses the HFRCO to generate a clock as a ratio of a reference clock source. It provides the following features:

- Frequency-lock mode. Only the output frequency is controlled, phase error is allowed to accumulate between the output and reference clock.
- Phase-lock mode. Both the output frequency and phase are controlled.
- Output frequency = FREF*(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.
- Output spectrum-spreading. The DPLL can randomize the generated output period by a configurable amount of spread.

It is important to note that when DPLL is enabled, the HFRCO output frequency will be generated according to the DPLL configuration.

10.3.12.1 Enabling and Disabling

The DPLL feature can be enabled and disabled by software via the CMU_OSCENCMD register. The FINETUNINGEN bit in the CMU_HFRCOCTRL must also be set for proper DPLL operation. When enabled, the DPLL feature controls the output frequency of the HFRCO. Before enabling DPLL, all clock muxes selecting HFRCO should be switched to HFRCODIV2 temporarily until the DPLL is locked, to avoid over-clocking the circuit. After DPLL is enabled and running, the clock muxes may be switched back to HFRCO to use the new output frequency. The DPLL is disabled automatically when entering EM2, EM3, or EM4.

10.3.12.2 Lock Modes

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. But the phase error can accumulate over time and cause the average frequency error non-zero. FREQLL mode also provide 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 and make the average frequency error zero. FREQLL mode should be used unless specific phase requirement exists.

10.3.12.3 Configurations

Output frequency = FREF*(N+1)/(M+1). User should calculate N and M to achieve the target frequency. Note that with N increases, the DCO lock time would increase and DCO jitter would decrease. Both directions 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=299, M=99} for fast lock time but high jitter, or as {N=2999, M=999} for lower jitter but longer lock time. For a good balance, N is suggested to be larger than 300 unless specific lock time is required.

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

10.3.12.4 Lock Detection

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

- If DPLLRDY interrupt is received first, it means target clock is ready and it is safe to switch to use DCO's output.
- If DPLLLOCKFAILLOW interrupt is received first, it indicates the RANGE in CMU_HFRCOCTRL is too small. User should disable DPLL and write a larger value to RANGE, then enable DPLL again to lock.
- If DPLLLOCKFAILHIGH interrupt is received first, it indicates the RANGE in CMU_HFRCOCTRL is too large. User should disable DPLL and write a smaller value to RANGE, then enable DPLL again to lock.
- If DPLLRDY interrupt is received first and then DPLLLOCKFAILLOW or DPLLLOCKFAILHIGH is received later, it means reference clock drifted over 2% and made DPLL lost its locked status.
  - If AUTORECOVER in CMU_DPLLCTRL is not set, user should disable DPLL and enable DPLL again to lock.
  - If AUTORECOVER in CMU_DPLLCTRL is set, hardware would 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 would still be received. User should disable DPLL and modify RANGE in CMU_HFRCOCTRL in corresponding direction like the second and third cases. Then enable DPLL again to lock.

10.3.12.5 Spectrum Spreading

Spreading of the DCO output spectrum is accomplished by driving a dedicated 5-bit DCO trim control with a digitally-generated, pseudo-random value. A centered, uniform, random spreading algorithm was selected.The spectrum-spreading pattern is generated by a single 10-bit linear feedback shift register (LFSR). To avoid high correlation between nearby values, a 5-step leap-forward LFSR update method is employed in the DPLL. The DCO output period can be randomized with a peak-to-peak amplitude given approximately by: 2^((SSAMP-1))*0.2%, where SSAMP is a register field in CMU_HFRCOSS. The generated random values are applied at regular intervals given by: 4*T_DCO*(SSINV+1), where T_DCO is DCO period and SSINV is a register field in CMU_HFRCOSS.

10.3.13 USB Clock Recovery

The USB clock recovery feature uses an active USB connection to tune the USHFRCO to 48 MHz, with accuracy suitable for full-speed and low-speed USB devices (USB clock recovery is not possible in host applications). This eliminates the need for a precise 48 MHz crystal in USB device applications. To configure USB clock recovery on the USHFRCO, the following steps should be performed:

1. Write the calibrated value for USHFRCO 48 MHz to CMU_USHFRCOCTRL. The 48 MHz calibration value is found in the USHFRCOCAL13 value stored in the DI page of flash.

   Note: From any device reset, the calibrated value is already populated in the CMU_USHFRCOCTRL register. If software has not adjusted this register for any other purpose, this write is not necessary.

2. Enable USB clock recovery in the CMU_USBCRCTRL register:
   a. For full speed device applications, write USBCREN to 1 and USBLSCRMD to 0.
   b. For low speed device applications, write USBCREN and USBLSCRMD both to 1.
3. Configure the USB rate clock to use USHFRCO by writing USBCLKSEL in CMU_USBCTRL to USHFRCO.
4. Enable the USB rate clock by setting USBCLKEN in CMU_USBCTRL to 1.
5. Enable the USHFRCO by setting the USHFRCOEN bit to 1 in CMU_OSCENMD.

It is important to note that FINETUNINGEN must be set to 1 when using USB clock recovery. Changing the setting of FINETUNINGEN will result in a frequency shift.

10.4 Register Map

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

Offset	Name	Type	Description
0x000	CMU_CTRL	RW	CMU Control Register
0x008	CMU_USHFRCOCTRL	RWH	USHFRCO Control Register
0x010	CMU_HFRCOCTRL	RWH	HFRCO Control Register
0x018	CMU_AUXHFRCOCTRL	RW	AUXHFRCO Control Register
0x020	CMU_LFRCOCTRL	RW	LFRCO Control Register
0x024	CMU_HFXOCTRL	RW	HFXO Control Register
0x028	CMU_HFXOCTRL1	RW	HFXO Control 1
0x02C	CMU_HFXOSTARTUPCTRL	RW	HFXO Startup Control
0x030	CMU_HFXOSTEADYSTATECTRL	RW	HFXO Steady State Control
0x034	CMU_HFXOTIMEOUTCTRL	RW	HFXO Timeout Control
0x038	CMU_LFXOCTRL	RW	LFXO Control Register
0x040	CMU_DPLLCTRL	RW	DPLL Control Register
0x044	CMU_DPLLCTRL1	RW	DPLL Control Register
0x050	CMU_CALCTRL	RW	Calibration Control Register
0x054	CMU_CALCNT	RWH	Calibration Counter Register
0x060	CMU_OSCENCMD	W1	Oscillator Enable/Disable Command Register
0x064	CMU_CMD	W1	Command Register
0x070	CMU_DBGCLKSEL	RW	Debug Trace Clock Select
0x074	CMU_HFCLKSEL	W1	High Frequency Clock Select Command Register
0x080	CMU_LFACLKSEL	RW	Low Frequency A Clock Select Register
0x084	CMU_LFBCLKSEL	RW	Low Frequency B Clock Select Register
0x088	CMU_LFECLKSEL	RW	Low Frequency E Clock Select Register
0x08C	CMU_LFCCLKSEL	RW	Low Frequency C Clock Select Register
0x090	CMU_STATUS	R	Status Register
0x094	CMU_HFCLKSTATUS	R	HFCLK Status Register
0x09C	CMU_HFXOTRIMSTATUS	R	HFXO Trim Status
0x0A0	CMU_IF	R	Interrupt Flag Register
0x0A4	CMU_IFS	W1	Interrupt Flag Set Register
0x0A8	CMU_IFC	(R)W1	Interrupt Flag Clear Register
0x0AC	CMU_IEN	RW	Interrupt Enable Register
0x0B0	CMU_HFBUSCLKEN0	RW	High Frequency Bus Clock Enable Register 0
0x0C0	CMU_HFPERCLKEN0	RW	High Frequency Peripheral Clock Enable Register 0
0x0C4	CMU_HFPERCLKEN1	RW	High Frequency Peripheral Clock Enable Register 1
0x0E0	CMU_LFACLKEN0	RW	Low Frequency a Clock Enable Register 0 (Async Reg)
0x0E8	CMU_LFBCLKEN0	RW	Low Frequency B Clock Enable Register 0 (Async Reg)

Offset	Name	Type	Description
0x0EC	CMU_LFCCLKEN0	RW	Low Frequency C Clock Enable Register 0 (Async Reg)
0x0F0	CMU_LFECLKEN0	RW	Low Frequency E Clock Enable Register 0 (Async Reg)
0x100	CMU_HFPRESC	RW	High Frequency Clock Prescaler Register
0x104	CMU_HFBUSPRESC	RW	High Frequency Bus Clock Prescaler Register
0x108	CMU_HFCOREPRESC	RW	High Frequency Core Clock Prescaler Register
0x10C	CMU_HFPERPRESC	RW	High Frequency Peripheral Clock Prescaler Register
0x114	CMU_HFEXPPRESC	RW	High Frequency Export Clock Prescaler Register
0x118	CMU_HFPERPRESCB	RW	High Frequency Peripheral Clock Prescaler B Register
0x11C	CMU_HFPERPRESCC	RW	High Frequency Peripheral Clock Prescaler C Register
0x120	CMU_LFAPRESC0	RW	Low Frequency a Prescaler Register 0 (Async Reg)
0x128	CMU_LFBPRESC0	RW	Low Frequency B Prescaler Register 0 (Async Reg)
0x130	CMU_LFEPRESC0	RW	Low Frequency E Prescaler Register 0 (Async Reg)
0x140	CMU_SYNCBUSY	R	Synchronization Busy Register
0x144	CMU_FREEZE	RW	Freeze Register
0x150	CMU_PCNTCTRL	RWH	PCNT Control Register
0x15C	CMU_ADCCTRL	RWH	ADC Control Register
0x160	CMU_SDIOCTRL	RW	SDIO Control Register
0x164	CMU_QSPICTRL	RW	QSPI Control Register
0x168	CMU_PDMCTRL	RW	PDM Control Register
0x170	CMU_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x174	CMU_ROUTELOC0	RW	I/O Routing Location Register
0x178	CMU_ROUTELOC1	RW	I/O Routing Location Register
0x180	CMU_LOCK	RWH	Configuration Lock Register
0x184	CMU_HFRCOSS	RW	HFRCO Spread Spectrum Register
0x1F0	CMU_USBCTRL	RWH	USB Control Register
0x1F4	CMU_USBCRCTRL	RW	USB Clock Recovery Control

10.5 Register Description

10.5.1 CMU_CTRL - CMU 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												1				0		0x00					0x00					0x00				
Access												RW				RW		RW					RW					RW				
Name												HFPERCLKEN				WSHFLE		CLKOUTSEL2					CLKOUTSEL1					CLKOUTSEL0				

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	HFPERCLKEN	1	RW	**HFPERCLK Enable**
	Set to enable the HFPERCLK.			
19:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	WSHFLE	0	RW	**Wait State for High-Frequency LE Interface**
	Set to allow access to LE peripherals when running $HFBUSCLK_{LE}$ at frequencies higher than 32 MHz			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:10	CLKOUTSEL2	0x00	RW	**Clock Output Select 2**
	Controls the clock output 2 multiplexer. To actually output on the pin, set CLKOUT2PEN in CMU_ROUTE.			

Value	Mode	Description
0	DISABLED	Disabled
1	ULFRCO	ULFRCO (directly from oscillator)
2	LFRCO	LFRCO (directly from oscillator)
3	LFXO	LFXO (directly from oscillator)
5	HFXODIV2Q	HFXO divided by two (qualified)
6	HFXO	HFXO (directly from oscillator)
7	HFEXPCLK	HFEXPCLK
8	HFXOX2Q	HFXO doubler (qualified) (doubling activated by HFXOX2EN=1)
9	ULFRCOQ	ULFRCO (qualified)
10	LFRCOQ	LFRCO (qualified)
11	LFXOQ	LFXO (qualified)
12	HFRCOQ	HFRCO (qualified)
13	AUXHFRCOQ	AUXHFRCO (qualified)

Bit	Name	Reset	Access	Description
	14	HFXOQ		HFXO (qualified)
	15	HFSRCCLK		HFSRCCLK
	18	USHFRCOQ		USHFRCO (qualified)
9:5	CLKOUTSEL1	0x00	RW	**Clock Output Select 1**

Controls the clock output 1 multiplexer. To actually output on the pin, set CLKOUT1PEN in CMU_ROUTE.

Value	Mode	Description
0	DISABLED	Disabled
1	ULFRCO	ULFRCO (directly from oscillator)
2	LFRCO	LFRCO (directly from oscillator)
3	LFXO	LFXO (directly from oscillator)
6	HFXO	HFXO (directly from oscillator)
7	HFEXPCLK	HFEXPCLK
9	ULFRCOQ	ULFRCO (qualified)
10	LFRCOQ	LFRCO (qualified)
11	LFXOQ	LFXO (qualified)
12	HFRCOQ	HFRCO (qualified)
13	AUXHFRCOQ	AUXHFRCO (qualified)
14	HFXOQ	HFXO (qualified)
15	HFSRCCLK	HFSRCCLK
18	USHFRCOQ	USHFRCO (qualified)

4:0	CLKOUTSEL0	0x00	RW	**Clock Output Select 0**

Controls the clock output multiplexer. To actually output on the pin, set CLKOUT0PEN in CMU_ROUTE.

Value	Mode	Description
0	DISABLED	Disabled
1	ULFRCO	ULFRCO (directly from oscillator)
2	LFRCO	LFRCO (directly from oscillator)
3	LFXO	LFXO (directly from oscillator)
6	HFXO	HFXO (directly from oscillator)
7	HFEXPCLK	HFEXPCLK
9	ULFRCOQ	ULFRCO (qualified)
10	LFRCOQ	LFRCO (qualified)
11	LFXOQ	LFXO (qualified)
12	HFRCOQ	HFRCO (qualified)
13	AUXHFRCOQ	AUXHFRCO (qualified)
14	HFXOQ	HFXO (qualified)
15	HFSRCCLK	HFSRCCLK

Bit	Name	Reset	Access	Description
	18	USHFRCOQ		USHFRCO (qualified)

10.5.2 CMU_USHFRCOCTRL - USHFRCO Control Register

Write this register to set the frequency band in which the USHFRCO is to operate. Always update all fields in this register at once by writing the value for the desired band, which has been obtained from the Device Information page entry for that band. The TUNING, FINETUNING, FINETUNINGEN and CLKDIV bitfields can be used to tune a specific band (FREQRANGE) of the oscillator to a non-preconfigured frequency. When changing this setting there will be no glitches on the USHFRCO output, hence it is safe to change this

setting even while the system is running on the USHFRCO. Only write CMU_USHFRCOCTRL when it is ready for an update as indicated by USHFRCOBSY=0 in CMU_SYNCBUSY.

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	0xB				0	0x0		1	0x2			0x08							0x1F							0x7F					
Access	RW				RW	RW		RW	RW			RW							RWH							RW					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Reset	Access	Description
31:28	VREFTC	0xB	RW	**USHFRCO Temperature Coefficient Trim on Comparator Reference**
	Writing this field adjusts the temperature coefficient trim on comparator reference.			
27	FINETUNINGEN	0	RW	**Enable Reference for Fine Tuning**
	Settings this bit enables USHFRCO fine tuning.			
26:25	CLKDIV	0x0	RW	**Locally Divide USHFRCO Clock Output**
	Writing this field configures the USHFRCO clock output divider.			

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

24	LDOHP	1	RW	**USHFRCO LDO High Power Mode**
	Settings this bit puts the USHFRCO LDO in high power mode.			
23:21	CMPBIAS	0x2	RW	**USHFRCO Comparator Bias Current**
	Writing this field adjusts the USHFRCO comparator bias current.			
20:16	FREQRANGE	0x08	RW	**USHFRCO Frequency Range**
	Writing this field adjusts the USHFRCO frequency range.			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	FINETUNING	0x1F	RWH	**USHFRCO Fine Tuning Value**
	Writing this field adjusts the USHFRCO fine tuning value. Higher value means lower frequency. Fine tuning is only enabled when FINETUNINGEN is set.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	TUNING	0x7F	RW	**USHFRCO Tuning Value**
	Writing this field adjusts the USHFRCO tuning value. Higher value means lower frequency.			

10.5.3 CMU_HFRCOCTRL - HFRCO Control Register

Write this register to set the frequency band in which the HFRCO is to operate. Always update all fields in this register at once by writing the value for the desired band, which has been obtained from the Device Information page entry for that band. The TUNING, FINETUNING, FINETUNINGEN and CLKDIV bitfields can be used to tune a specific band (FREQRANGE) of the oscillator to a non-preconfigured frequency. When changing this setting there will be no glitches on the HFRCO output, hence it is safe to change this setting

even while the system is running on the HFRCO. Only write CMU_HFRCOCTRL when it is ready for an update as indicated by HFRCOBSY=0 in CMU_SYNCBUSY.

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	0xB				0	0x0		1	0x2			0x08							0x1F							0x7F					
Access	RWH				RWH	RWH		RWH	RWH			RWH							RWH							RWH					
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING					

Bit	Name	Reset	Access	Description
31:28	VREFTC	0xB	RWH	**HFRCO Temperature Coefficient Trim on Comparator Reference**
	Writing this field adjusts the temperature coefficient trim on comparator reference.			
27	FINETUNINGEN	0	RWH	**Enable Reference for Fine Tuning**
	Settings this bit enables HFRCO fine tuning.			
26:25	CLKDIV	0x0	RWH	**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.

24	LDOHP	1	RWH	**HFRCO LDO High Power Mode**
	Settings this bit puts the HFRCO LDO in high power mode.			
23:21	CMPBIAS	0x2	RWH	**HFRCO Comparator Bias Current**
	Writing this field adjusts the HFRCO comparator bias current.			
20:16	FREQRANGE	0x08	RWH	**HFRCO Frequency Range**
	Writing this field adjusts the HFRCO frequency range.			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	FINETUNING	0x1F	RWH	**HFRCO Fine Tuning Value**
	Writing this field adjusts the HFRCO fine tuning value. Higher value means lower frequency. Fine tuning is only enabled when FINETUNINGEN is set.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	TUNING	0x7F	RWH	**HFRCO Tuning Value**
	Writing this field adjusts the HFRCO tuning value. Higher value means lower frequency.			

10.5.4 CMU_AUXHFRCOCTRL - AUXHFRCO Control Register

Write this register with the production calibrated values from the Device Info pages. The TUNING, FINETUNING, FINETUNINGEN and CLKDIV bitfields can be used to tune a specific band (FREQRANGE) of the oscillator to a non-preconfigured frequency. Only write CMU_AUXHFRCOCTRL when it is ready for an update as indicated by AUXHFRCOBSY=0 in CMU_SYNCBUSY.

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	0xB				0	0x0		1	0x2			0x08							0x1F							0x7F						
Access	RW				RW	RW		RW	RW			RW							RW							RW						
Name	VREFTC				FINETUNINGEN	CLKDIV		LDOHP	CMPBIAS			FREQRANGE							FINETUNING							TUNING						

Bit	Name	Reset	Access	Description
31:28	VREFTC	0xB	RW	**AUXHFRCO Temperature Coefficient Trim on Comparator Reference**
	Writing this field adjusts the temperature coefficient trim on comparator reference.			
27	FINETUNINGEN	0	RW	**Enable Reference for Fine Tuning**
	Settings this bit enables AUXHFRCO fine tuning.			
26:25	CLKDIV	0x0	RW	**Locally Divide AUXHFRCO Clock Output**
	Writing this field configures the AUXHFRCO clock output divider.			

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

24	LDOHP	1	RW	**AUXHFRCO LDO High Power Mode**
	Settings this bit puts the AUXHFRCO LDO in high power mode.			
23:21	CMPBIAS	0x2	RW	**AUXHFRCO Comparator Bias Current**
	Writing this field adjusts the AUXHFRCO comparator bias current.			
20:16	FREQRANGE	0x08	RW	**AUXHFRCO Frequency Range**
	Writing this field adjusts the AUXHFRCO frequency range.			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	FINETUNING	0x1F	RW	**AUXHFRCO Fine Tuning Value**
	Writing this field adjusts the AUXHFRCO fine tuning value. Higher value means lower frequency. Fine tuning is only enabled when FINETUNINGEN is set.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	TUNING	0x7F	RW	**AUXHFRCO Tuning Value**
	Writing this field adjusts the AUXHFRCO tuning value. Higher value means lower frequency.			

10.5.5 CMU_LFRCOCTRL - LFRCO 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	0x8						0x1				0x1			1	1	0								0x100								
Access	RW						RW				RW			RW	RW	RW								RW								
Name	GMCCURTUNE						TIMEOUT				VREFUPDATE			ENDEM	ENCHOP	ENVREF								TUNING								

Bit	Name	Reset	Access	Description
31:28	GMCCURTUNE	0x8	RW	**Tuning of Gmc Current**
	Set to tune GMC current. This field is updated with the production calibrated value during reset, and the reset value might therefore vary between devices.			
27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:24	TIMEOUT	0x1	RW	**LFRCO Timeout**
	Configures the start-up delay for LFRCO. Do not change while LFRCO is enabled. When starting up the LFRCO after it has been completely turned off, use TIMEOUT=16cycles. If the LFRCO has been retained on in EM4, then the TIMEOUT=2cycles configuration is also allowed when re-enabling the LFRCO after EM4 exit (as it is still running).			

Value	Mode	Description
0	2CYCLES	Timeout period of 2 cycles
1	16CYCLES	Timeout period of 16 cycles
2	32CYCLES	Timeout period of 32 cycles

23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	VREFUPDATE	0x1	RW	**Control Vref Update Rate**
	Specify Vref update rate. This field can be updated with the production test value during reset, and the reset value might therefore differ.			

Value	Mode	Description
0	32CYCLES	32 clocks.
1	64CYCLES	64 clocks.
2	128CYCLES	128 clocks.
3	256CYCLES	256 clocks.

19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	ENDEM	1	RW	**Enable Dynamic Element Matching**
	Set to enable dynamic element matching. This improves average frequency accuracy at the cost of increased jitter.			

Bit	Name	Reset	Access	Description
17	ENCHOP	1	RW	**Enable Comparator Chopping**
	Set to enable comparator chopping. This improves average frequency accuracy at the cost of increased jitter.			
16	ENVREF	0	RW	**Enable Duty Cycling of Vref**
	Set to enable duty cycling of vref. Clear during calibration of LFRCO. Only change when LFRCO is off.			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	TUNING	0x100	RW	**LFRCO Tuning Value**
	Writing this field adjusts the LFRCO frequency (the higher the value, the lower the frequency). This field is updated with the production calibrated value during reset, and the reset value might therefore vary between devices.			

10.5.6 CMU_HFXOCTRL - HFXO Control 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			0	0		0x0																					0x0		1		0x0	
Access			RW	RW		RW																					RW		RW		RW	
Name			AUTOSTARTSELEM0EM1	AUTOSTARTEM0EM1		LFTIMEOUT																					PEAKDETMODE		HFXOX2EN		MODE	

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	AUTOSTARTSELEM0EM1	0	RW	**Automatically Start and Select of HFXO Upon EM0/EM1 Entry From EM2/EM3**
	This bit enables automatic start-up and immediate selection of the HFXO when in EM0/EM1 (also after entry from EM2/EM3). Note that setting this bit to 1 will stall HFSRCCLK until HFXO becomes ready. Allowed to change at any time.			
28	AUTOSTARTEM0EM1	0	RW	**Automatically Start of HFXO Upon EM0/EM1 Entry From EM2/EM3**
	This bit enables automatic start-up of the HFXO when in EM0/EM1 (also after entry from EM2/EM3) without causing an automatic HFXO selection. Allowed to change at any time.			
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:24	LFTIMEOUT	0x0	RW	**HFXO Low Frequency Timeout**
	Configures the start-up delay for HFXO measured in LFECLK cycles. Only change when both HFXO and LFECLK are off.			

Value	Mode	Description
0	0CYCLES	Timeout period of 0 cycles (disabled)
1	2CYCLES	Timeout period of 2 cycles
2	4CYCLES	Timeout period of 4 cycles
3	16CYCLES	Timeout period of 16 cycles
4	32CYCLES	Timeout period of 32 cycles
5	64CYCLES	Timeout period of 64 cycles
6	1KCYCLES	Timeout period of 1024 cycles
7	4KCYCLES	Timeout period of 4096 cycles

23:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
5:4	PEAKDETMODE	0x0	RW	**HFXO Automatic Peak Detection Mode**

Set to AUTOCMD to allow automatic HFXO peak detection (MANUAL mode provides direct control of IBTRIMXOCORE and PEAKDETEN).

Value	Mode	Description
0	ONCECMD	Automatic control of HFXO peak detection sequence. Only performs peak detection on initial HFXO startup. CMU_CMD HFXOPEAKDETSTART allowed to be used after HFXORDY=1.
1	AUTOCMD	Automatic control of HFXO peak detection sequence. CMU_CMD HFXOPEAKDETSTART allowed to be used after HFXORDY=1.
2	CMD	CMU_CMD HFXOPEAKDETSTART can be used to trigger the peak detection sequence after HFXORDY=1.
3	MANUAL	CMU_HFXOSTEADYSTATECTRL IBTRIMXOCORE and PEAKDETEN are under full software control and are allowed to be changed once HFXO is ready.

3	HFXOX2EN	1	RW	**Enable Double Frequency on HFXOX2 Clock (compared to HFXO Clock)**

Set to clock doubler on HFXOX2. If not set, then HFXOX2 frequency will equal HFXO frequency. Only allowed for XTALs up to 25 MHz

2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	MODE	0x0	RW	**HFXO Mode**

Set this to configure the external source for the HFXO. The oscillator setting takes effect when 1 is written to HFXOEN in CMU_OSCENCMD.

Value	Mode	Description
0	XTAL	4 MHz - 50 MHz crystal oscillator
1	ACBUFEXTCLK	An AC coupled buffer is coupled in series with HFXTAL_N pin, suitable for external sinus wave.
2	DCBUFEXTCLK	A DC coupled buffer is coupled in series with HFXTAL_N pin, suitable for external sinus wave.
3	DIGEXTCLK	Digital external clock can be supplied on HFXTAL_N pin.

10.5.7 CMU_HFXOCTRL1 - HFXO Control 1

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																			0x2													
Access																			RW													
Name																			PEAKDETTHR													

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*		
14:12	PEAKDETTHR	0x2	RW	**Sets the Amplitude Detection Level (mV)**

Configures the Peak Detection threshold. It is not allowed to change when hardware based Peak Detection Algorithm or Peak Monitoring Algorithm is being performed. Allowed to change when HFXOBSY=0. Allowed to change after completing automatic Peak Detection (HFXOBSY=1, PEAKMONEN=0, HFXOPEAKDETRDY=1). Allowed to change after completing HFXO startup when not using automatic Peak Detection (HFXOBSY=1, PEAKMONEN=0, HFXORDY=1)

Value	Mode	Description
0	THR0	50mV amplitude detection level
1	THR1	75mV amplitude detection level
2	THR2	115mV amplitude detection level
3	THR3	160mV amplitude detection level
4	THR4	220mV amplitude detection level
5	THR5	260mV amplitude detection level
6	THR6	320mV amplitude detection level
7	THR7	Same as THR6

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

10.5.8 CMU_HFXOSTARTUPCTRL - HFXO Startup Control

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													0x000									0x600										
Access													RW									RW										
Name													CTUNE									IBTRIMXOCORE										

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*		
19:11	CTUNE	0x000	RW	**Sets Oscillator Tuning Capacitance** This CTUNE value is applied during the startup phase of the HFXO. The required CTUNE value is XTAL specific. Capacitance on HFXTAL_N and HFXTAL_P (pF) = C_tune = Min (C_HFXO_T) + CTUNE<8:0> x SS_HFXO. Please find C_HFXO_T and SS_HFXO in the datasheet.
10:0	IBTRIMXOCORE	0x600	RW	**Sets the Startup Oscillator Core Bias Current** This IBTRIMXOCORE value is applied during the startup phase of the HFXO. Current (uA) = IBTRIMXOCORE<10:9> X 1280uA + IBTRIMXOCORE<8:0> X 2uA. It is recommended to use IBTRIMXOCORE<8:0>=0.

10.5.9 CMU_HFXOSTEADYSTATECTRL - HFXO Steady State 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					1	0											0x000										0x100					
Access					RW	RW											RW										RW					
Name					PEAKMONEN	PEAKDETEN											CTUNE										IBTRIMXOCORE					

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	PEAKMONEN	1	RW	**Automatically Perform Peak Monitoring Algorithm on Every Rising Edge of ULFRCO** This bit enables Peak Monitoring Algorithm to be performed on every rising edge of ULFRCO. Allowed to change at any time.
26	PEAKDETEN	0	RW	**Enables Oscillator Peak Detectors** Direct control allowed after completion of automatic Peak Detection (PEAKDETRDY=1) or when HFXO is ready and automatic Peak Detection is not being used (HFXORDY=1).
25:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:11	CTUNE	0x000	RW	**Sets Oscillator Tuning Capacitance** This CTUNE value is applied during the steady state phase of the HFXO (as well as during the peak detection and shunt current optimization algorithms). Direct control is allowed when HFXORDY=1. The required CTUNE value is XTAL specific. Capacitance on HFXTAL_N and HFXTAL_P (pF) = C_tune = Min (C_HFXO_T) + CTUNE<8:0> x SS_HFXO. Please find C_HFXO_T and SS_HFXO in the datasheet.
10:0	IBTRIMXOCORE	0x100	RW	**Sets the Steady State Oscillator Core Bias Current.** This IBTRIMXOCORE value is applied during the steady state phase of the HFXO. Required IBTRIMXOCORE is XTAL specific. It is also used as the initial value during the peak detection algorithm. Direct control allowed when PEAKDETSHUNTOPTMODE=MANUAL and HFXO is ready. Current (uA) = IBTRIMXOCORE<10:9> x 1280uA + IBTRIMXOCORE<8:0> x 2uA. It is recommended to use IBTRIMXOCORE<10:9>=0..

10.5.10 CMU_HFXOTIMEOUTCTRL - HFXO Timeout Control

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																	0xD								0x8				0xE			
Access																	RW								RW				RW			
Name																	PEAKDETTIMEOUT								STEADYTIMEOUT				STARTUPTIMEOUT			

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	PEAKDETTIMEOUT	0xD	RW	**Wait Duration in HFXO Peak Detection Wait State**

Wait duration depends on the chosen XTAL (expected value is between 25 us and 200 us). Program the desired duration measured in cycles of (at least) 83 ns.

Value	Mode	Description
0	2CYCLES	Timeout period of 2 cycles
1	4CYCLES	Timeout period of 4 cycles
2	16CYCLES	Timeout period of 16 cycles
3	32CYCLES	Timeout period of 32 cycles
4	64CYCLES	Timeout period of 64 cycles
5	128CYCLES	Timeout period of 128 cycles
6	256CYCLES	Timeout period of 256 cycles
7	1KCYCLES	Timeout period of 1024 cycles
8	2KCYCLES	Timeout period of 2048 cycles
9	4KCYCLES	Timeout period of 4096 cycles
10	8KCYCLES	Timeout period of 8192 cycles
11	16KCYCLES	Timeout period of 16384 cycles
12	32KCYCLES	Timeout period of 32768 cycles
13	64KCYCLES	Timeout period of 65536 cycles
14	128KCYCLES	Timeout period of 131072 cycles

11:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:4	STEADYTIMEOUT	0x8	RW	**Wait Duration in HFXO Startup Steady Wait State**

Wait duration depends on the chosen XTAL (expected value is around 100 us). Program the desired duration measured in cycles of (at least) 83 ns.

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	2CYCLES		Timeout period of 2 cycles
	1	4CYCLES		Timeout period of 4 cycles
	2	16CYCLES		Timeout period of 16 cycles
	3	32CYCLES		Timeout period of 32 cycles
	4	64CYCLES		Timeout period of 64 cycles
	5	128CYCLES		Timeout period of 128 cycles
	6	256CYCLES		Timeout period of 256 cycles
	7	1KCYCLES		Timeout period of 1024 cycles
	8	2KCYCLES		Timeout period of 2048 cycles
	9	4KCYCLES		Timeout period of 4096 cycles
	10	8KCYCLES		Timeout period of 8192 cycles
	11	16KCYCLES		Timeout period of 16384 cycles
	12	32KCYCLES		Timeout period of 32768 cycles
	13	64KCYCLES		Timeout period of 65536 cycles
	14	128KCYCLES		Timeout period of 131072 cycles
3:0	STARTUPTIMEOUT	0xE	RW	**Wait Duration in HFXO Startup Enable Wait State**

Wait duration depends on the chosen XTAL (expected value is between 100 us and 1600 us). Program the desired duration measured in cycles of (at least) 83 ns.

Value	Mode	Description
0	2CYCLES	Timeout period of 2 cycles
1	4CYCLES	Timeout period of 4 cycles
2	16CYCLES	Timeout period of 16 cycles
3	32CYCLES	Timeout period of 32 cycles
4	64CYCLES	Timeout period of 64 cycles
5	128CYCLES	Timeout period of 128 cycles
6	256CYCLES	Timeout period of 256 cycles
7	1KCYCLES	Timeout period of 1024 cycles
8	2KCYCLES	Timeout period of 2048 cycles
9	4KCYCLES	Timeout period of 4096 cycles
10	8KCYCLES	Timeout period of 8192 cycles
11	16KCYCLES	Timeout period of 16384 cycles
12	32KCYCLES	Timeout period of 32768 cycles
13	64KCYCLES	Timeout period of 65536 cycles
14	128KCYCLES	Timeout period of 131072 cycles

10.5.11 CMU_LFXOCTRL - LFXO Control 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						0x7						0			0x0		1	0		0x2			0x0			0x00					
Access						RW						RW			RW		RW	RW		RW			RW			RW					
Name						TIMEOUT						BUFCUR			CUR		AGC	HIGHAMPL		GAIN			MODE			TUNING					

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	TIMEOUT	0x7	RW	**LFXO Timeout**

Configures the start-up delay for LFXO. Do not change while LFXO is enabled. When starting up the LFXO after it has been completely turned off, use the TIMEOUT setting required by the XTAL. If the LFXO has been retained on in EM4, then the TIMEOUT=2cycles configuration is also allowed when re-enabling the LFXO after EM4 exit (as it is still running).

Value	Mode	Description
0	2CYCLES	Timeout period of 2 cycles
1	256CYCLES	Timeout period of 256 cycles
2	1KCYCLES	Timeout period of 1024 cycles
3	2KCYCLES	Timeout period of 2048 cycles
4	4KCYCLES	Timeout period of 4096 cycles
5	8KCYCLES	Timeout period of 8192 cycles
6	16KCYCLES	Timeout period of 16384 cycles
7	32KCYCLES	Timeout period of 32768 cycles

23:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	BUFCUR	0	RW	**LFXO Buffer Bias Current**

The default value is intended to cover all use cases and reprogramming is not recommended. Do not change while LFXO is enabled.

19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	CUR	0x0	RW	**LFXO Current Trim**

The default value is intended to cover all use cases and reprogramming is not recommended. Do not change while LFXO is enabled.

15	AGC	1	RW	**LFXO AGC Enable**

Set this bit to enable automatic gain control which limits XTAL oscillation amplitude. Do not change while LFXO is enabled.

14	HIGHAMPL	0	RW	**LFXO High XTAL Oscillation Amplitude Enable**

Set this bit to enable high XTAL oscillation amplitude. Do not change while LFXO is enabled.

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

Bit	Name	Reset	Access	Description
12:11	GAIN	0x2	RW	**LFXO Startup Gain**
	The optimal value for maximum startup margin depends on the chosen XTAL. Refer to the device data sheet or Simplicity Studio for more information.			
10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	MODE	0x0	RW	**LFXO Mode**

Set this to configure the external source for the LFXO. Do not change while LFXO is enabled. The oscillator setting takes effect when 1 is written to LFXOEN in CMU_OSCENCMD. The oscillator setting is reset to default when 1 is written to LFXODIS in CMU_OSCENCMD.

Value	Mode	Description
0	XTAL	32768 Hz crystal oscillator
1	BUFEXTCLK	An AC coupled buffer is coupled in series with LFXTAL_N pin, suitable for external sinus wave (32768 Hz).
2	DIGEXTCLK	Digital external clock on LFXTAL_N pin. Oscillator is effectively bypassed.

7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	TUNING	0x00	RW	**LFXO Internal Capacitor Array Tuning Value**

Writing this field adjusts the internal load capacitance connected between LFXTAL_P and ground and LFXTAL_N and ground symmetrically (the higher the value, the higher the capacitance, the lower the frequency). Only increment or decrement by 1 LSB at a time.

10.5.12 CMU_DPLLCTRL - DPLL Control 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																										0		0x0		0	0	0
Access																										RW		RW		RW	RW	RW
Name																										DITHEN		REFSEL		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	0	RW	**Dither Enable Control**
	Set to enable the dither function			
5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4:3	REFSEL	0x0	RW	**Reference Clock Selection Control**
	This field selects which clock as the reference clock			

Value	Mode	Description
0	HFXO	HFXO selected
1	LFXO	LFXO selected
2	USHFRCO	USHFRCO selected
3	CLKIN0	CLKIN0 selected

Bit	Name	Reset	Access	Description
2	AUTORECOVER	0	RW	**Automatic Recovery Ctrl**
	Set to enable automatic recovery function.			
1	EDGESEL	0	RW	**Reference Edge Select**
	This bit controls which edge of reference is detected			

Value	Mode	Description
0	FALL	Falling edge
1	RISE	Rising edge

Bit	Name	Reset	Access	Description
0	MODE	0	RW	**Operating Mode Control**
	This bit controls which mode DPLL is operating when enabled			

Value	Mode	Description
0	FREQLL	DPLL operates in frequency-lock mode.
1	PHASELL	DPLL operates in phase-lock mode.

10.5.13 CMU_DPLLCTRL1 - DPLL Control 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					0x000																0x000											
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	0x000	RW	**Factor N**
	The locked DCO frequency is given by: Fdco = Fref * (N + 1) / (M + 1). N is required to be larger than 32.			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	M	0x000	RW	**Factor M**
	The locked DCO frequency is given by: Fdco = Fref * (N + 1) / (M + 1). M can be any value.			

10.5.14 CMU_CALCTRL - Calibration 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											0	0x0					0x0		
Access					RW								RW											RW	RW					RW		
Name					PRSDOWNSEL								PRSUPSEL											CONT	DOWNSEL					UPSEL		

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	PRSDOWNSEL	0x0	RW	**PRS Select for PRS Input When Selected in DOWNSEL**

Select PRS input for PRS based calibration. Only change when calibration circuit is off.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input
8	PRSCH8	PRS Channel 8 selected as input
9	PRSCH9	PRS Channel 9 selected as input
10	PRSCH10	PRS Channel 10 selected as input
11	PRSCH11	PRS Channel 11 selected as input
12	PRSCH12	PRS Channel 12 selected as input
13	PRSCH13	PRS Channel 13 selected as input
14	PRSCH14	PRS Channel 14 selected as input
15	PRSCH15	PRS Channel 15 selected as input

23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PRSUPSEL	0x0	RW	**PRS Select for PRS Input When Selected in UPSEL**

Select PRS input for PRS based calibration. Only change when calibration circuit is off.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input

Bit	Name	Reset	Access	Description
	1	PRSCH1		PRS Channel 1 selected as input
	2	PRSCH2		PRS Channel 2 selected as input
	3	PRSCH3		PRS Channel 3 selected as input
	4	PRSCH4		PRS Channel 4 selected as input
	5	PRSCH5		PRS Channel 5 selected as input
	6	PRSCH6		PRS Channel 6 selected as input
	7	PRSCH7		PRS Channel 7 selected as input
	8	PRSCH8		PRS Channel 8 selected as input
	9	PRSCH9		PRS Channel 9 selected as input
	10	PRSCH10		PRS Channel 10 selected as input
	11	PRSCH11		PRS Channel 11 selected as input
	12	PRSCH12		PRS Channel 12 selected as input
	13	PRSCH13		PRS Channel 13 selected as input
	14	PRSCH14		PRS Channel 14 selected as input
	15	PRSCH15		PRS Channel 15 selected as input
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	CONT	0	RW	**Continuous Calibration**
	Set this bit to enable continuous calibration			
7:4	DOWNSEL	0x0	RW	**Calibration Down-counter Select**
	Selects clock source for the calibration down-counter. Only change when calibration circuit is off.			
	Value	Mode		Description
	0	HFCLK		Select HFCLK for down-counter
	1	HFXO		Select HFXO for down-counter
	2	LFXO		Select LFXO for down-counter
	3	HFRCO		Select HFRCO for down-counter
	4	LFRCO		Select LFRCO for down-counter
	5	AUXHFRCO		Select AUXHFRCO for down-counter
	6	PRS		Select PRS input selected by PRSDOWNSEL as down-counter
	8	USHFRCO		Select USHFRCO for down-counter
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	UPSEL	0x0	RW	**Calibration Up-counter Select**
	Selects clock source for the calibration up-counter. Only change when calibration circuit is off.			
	Value	Mode		Description
	0	HFXO		Select HFXO as up-counter
	1	LFXO		Select LFXO as up-counter

Bit	Name	Reset	Access	Description
	2	HFRCO		Select HFRCO as up-counter
	3	LFRCO		Select LFRCO as up-counter
	4	AUXHFRCO		Select AUXHFRCO as up-counter
	5	PRS		Select PRS input selected by PRSUPSEL as up-counter
	7	USHFRCO		Select USHFRCO as up-counter

10.5.15 CMU_CALCNT - Calibration Counter 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													0x00000																			
Access													RWH																			
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	0x00000	RWH	**Calibration Counter**
	Write top value before calibration. Read calibration result from this register when Calibration Ready flag has been set.			

10.5.16 CMU_OSCENCMD - Oscillator Enable/Disable Command 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																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access																			W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																			DPLLDIS	DPLLEN	USHFRCODIS	USHFRCOEN	LFXODIS	LFXOEN	LFRCODIS	LFRCOEN	AUXHFRCODIS	AUXHFRCOEN	HFXODIS	HFXOEN	HFRCODIS	HFRCOEN

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	DPLLDIS	0	W1	**DPLL Disable**
	Disables the DPLL.			
12	DPLLEN	0	W1	**DPLL Enable**
	Enables the DPLL.			
11	USHFRCODIS	0	W1	**USHFRCO Disable**
	Disables the USHFRCO. USHFRCOEN has higher priority if written simultaneously.			
10	USHFRCOEN	0	W1	**USHFRCO Enable**
	Enables the USHFRCO.			
9	LFXODIS	0	W1	**LFXO Disable**
	Disables the LFXO. LFXOEN has higher priority if written simultaneously. WARNING: Do not disable the LFXO if this oscillator is selected as the source for HFCLK. When waking up from EM4 make sure EM4UNLATCH in EMU_CMD is set for this to take effect			
8	LFXOEN	0	W1	**LFXO Enable**
	Enables the LFXO. When waking up from EM4 make sure EM4UNLATCH in EMU_CMD is set for this to take effect			
7	LFRCODIS	0	W1	**LFRCO Disable**
	Disables the LFRCO. LFRCOEN has higher priority if written simultaneously. WARNING: Do not disable the LFRCO if this oscillator is selected as the source for HFCLK. When waking up from EM4 make sure EM4UNLATCH in EMU_CMD is set for this to take effect			
6	LFRCOEN	0	W1	**LFRCO Enable**
	Enables the LFRCO. When waking up from EM4 make sure EM4UNLATCH in EMU_CMD is set for this to take effect			
5	AUXHFRCODIS	0	W1	**AUXHFRCO Disable**
	Disables the AUXHFRCO. AUXHFRCOEN has higher priority if written simultaneously.			
4	AUXHFRCOEN	0	W1	**AUXHFRCO Enable**
	Enables the AUXHFRCO.			
3	HFXODIS	0	W1	**HFXO Disable**
	Disables the HFXO. HFXOEN has higher priority if written simultaneously. WARNING: Do not disable the HFXO if this oscillator is selected as the source for HFCLK.			

Bit	Name	Reset	Access	Description
2	HFXOEN	0	W1	**HFXO Enable**
	Enables the HFXO.			
1	HFRCODIS	0	W1	**HFRCO Disable**
	Disables the HFRCO. HFRCOEN has higher priority if written simultaneously. WARNING: Do not disable the HFRCO if this oscillator is selected as the source for HFCLK.			
0	HFRCOEN	0	W1	**HFRCO Enable**
	Enables the HFRCO.			

10.5.17 CMU_CMD - Command 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																												0			0	0
Access																												W1			W1	W1
Name																												HFXOPEAKDETSTART			CALSTOP	CALSTART

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	HFXOPEAKDET-START	0	W1	**HFXO Peak Detection Start**
	Starts the HFXO peak detection and runs it one time.			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	CALSTOP	0	W1	**Calibration Stop**
	Stops the calibration counters.			
0	CALSTART	0	W1	**Calibration Start**
	Starts the calibration, effectively loading the CMU_CALCNT into the down-counter and start decrementing.			

10.5.18 CMU_DBGCLKSEL - Debug Trace Clock Select

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																															DBG	

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	DBG	0x0	RW	**Debug Trace Clock**

Select clock used for debug trace.

Value	Mode	Description
0	AUXHFRCO	AUXHFRCO is the debug trace clock
1	HFCLK	HFCLK is the debug trace clock
2	HFRCODIV2	HFRCO divided by 2 is the debug trace clock

10.5.19 CMU_HFCLKSEL - High Frequency Clock Select Command 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																														W1		
Name																														HF		

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	HF	0x0	W1	**HFCLK Select**

Selects the clock source for HFCLK. Note that selecting an oscillator that is disabled will cause the system clock to stop. Check the status register and confirm that oscillator is ready before switching. If the system can deal with a temporarily stopped system clock, then it is okay to switch to an oscillator as soon as the status register indicates that the oscillator has been enabled successfully.

Value	Mode	Description
1	HFRCO	Select HFRCO as HFCLK
2	HFXO	Select HFXO as HFCLK
3	LFRCO	Select LFRCO as HFCLK
4	LFXO	Select LFXO as HFCLK
5	HFRCODIV2	Select HFRCO divided by 2 as HFCLK
6	USHFRCO	Select USHFRCO as HFCLK
7	CLKIN0	Select CLKIN0 as HFCLK

10.5.20 CMU_LFACLKSEL - Low Frequency A Clock Select 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																														LFA		

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	LFA	0x0	RW	**Clock Select for LFA**

Selects the clock source for LFACLK.

Value	Mode	Description
0	DISABLED	LFACLK is disabled
1	LFRCO	LFRCO selected as LFACLK
2	LFXO	LFXO selected as LFACLK
4	ULFRCO	ULFRCO selected as LFACLK

10.5.21 CMU_LFBCLKSEL - Low Frequency B Clock Select 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																														LFB		

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	LFB	0x0	RW	**Clock Select for LFB**

Selects the clock source for LFBCLK.

Value	Mode	Description
0	DISABLED	LFBCLK is disabled
1	LFRCO	LFRCO selected as LFBCLK
2	LFXO	LFXO selected as LFBCLK
3	HFCLKLE	HFCLK divided by two/four is selected as LFBCLK
4	ULFRCO	ULFRCO selected as LFBCLK

10.5.22 CMU_LFECLKSEL - Low Frequency E Clock Select Register

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																														LFE		

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	LFE	0x0	RW	**Clock Select for LFE**

Selects the clock source for LFECLK. When waking up from EM4 make sure EM4UNLATCH in EMU_CMD is set for this to take effect

Value	Mode	Description
0	DISABLED	LFECLK is disabled
1	LFRCO	LFRCO selected as LFECLK
2	LFXO	LFXO selected as LFECLK
4	ULFRCO	ULFRCO selected as LFECLK

10.5.23 CMU_LFCCLKSEL - Low Frequency C Clock Select Register

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																														RW		
Name																														LFC		

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	LFC	0x0	RW	**Clock Select for LFC**

Selects the clock source for LFCCLK.

Value	Mode	Description
0	DISABLED	LFCCLK is disabled
1	LFRCO	LFRCO selected as LFCCLK
2	LFXO	LFXO selected as LFCCLK
4	ULFRCO	ULFRCO selected as LFCCLK

10.5.24 CMU_STATUS - Status 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			0	0	0		0			0			0	0	0	1			0	0	0	0	0	0	0	0	0	0	0	0	1	1
Access			R	R	R		R			R			R	R	R	R			R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name			ULFRCOPHASE	LFRCOPHASE	LFXOPHASE		HFXOAMPLOW			HFXOPEAKDETRDY			PDMCLKENS	QSPI0CLKENS	SDIOCLKENS	CALRDY			DPLLRDY	DPLLENS	USHFRCORDY	USHFRCOENS	LFXORDY	LFXOENS	LFRCORDY	LFRCOENS	AUXHFRCORDY	AUXHFRCOENS	HFXORDY	HFXOENS	HFRCORDY	HFRCOENS

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	ULFRCOPHASE	0	R	**ULFRCO Clock Phase**
	Used to determine if ULFRCO is in high or low phase.			
28	LFRCOPHASE	0	R	**LFRCO Clock Phase**
	Used to determine if LFRCO is in high or low phase.			
27	LFXOPHASE	0	R	**LFXO Clock Phase**
	Used to determine if LFXO is in high or low phase.			
26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	HFXOAMPLOW	0	R	**HFXO Amplitude Tuning Value Too Low**
	HFXO oscillation amplitude is too low. When using PEAKDETSHUNTOPTMODE=MANUAL, the IBTRIMXOCORE value in CMU_HFXOSTEADYSTATECTRL should be tuned up by 1 LSB.			
24:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22	HFXOPEAKDETRDY	0	R	**HFXO Peak Detection Ready**
	HFXO peak detection is ready.			
21:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	PDMCLKENS	0	R	**PDM Clock Enabled Status**
	PDM clock is enabled (only valid if PDMCLKSEL is on a running clock source).			
18	QSPI0CLKENS	0	R	**QSPI0 Clock Enabled Status**
	QSPI0 clock is enabled (only valid if QSPI0CLKSEL is on a running clock source).			
17	SDIOCLKENS	0	R	**SDIO Clock Enabled Status**
	SDIO clock is enabled (only valid if SDIOCLKSEL is on a running clock source).			
16	CALRDY	1	R	**Calibration Ready**
	Calibration is Ready (0 when calibration is ongoing).			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
13	DPLLRDY	0	R	**DPLL Ready**
	DPLL is enabled and locked			
12	DPLLENS	0	R	**DPLL Enable Status**
	DPLL is enabled			
11	USHFRCORDY	0	R	**USHFRCO Ready**
	USHFRCO is enabled and start-up time has exceeded.			
10	USHFRCOENS	0	R	**USHFRCO Enable Status**
	USHFRCO is enabled.			
9	LFXORDY	0	R	**LFXO Ready**
	LFXO is enabled and start-up time has exceeded.			
8	LFXOENS	0	R	**LFXO Enable Status**
	LFXO is enabled (shows disabled status if EM4 repaint is required).			
7	LFRCORDY	0	R	**LFRCO Ready**
	LFRCO is enabled and start-up time has exceeded.			
6	LFRCOENS	0	R	**LFRCO Enable Status**
	LFRCO is enabled (shows disabled status if EM4 repaint is required).			
5	AUXHFRCORDY	0	R	**AUXHFRCO Ready**
	AUXHFRCO is enabled and start-up time has exceeded.			
4	AUXHFRCOENS	0	R	**AUXHFRCO Enable Status**
	AUXHFRCO is enabled.			
3	HFXORDY	0	R	**HFXO Ready**
	HFXO is enabled and start-up time has exceeded.			
2	HFXOENS	0	R	**HFXO Enable Status**
	HFXO is enabled.			
1	HFRCORDY	1	R	**HFRCO Ready**
	HFRCO is enabled and start-up time has exceeded.			
0	HFRCOENS	1	R	**HFRCO Enable Status**
	HFRCO is enabled.			

10.5.25 CMU_HFCLKSTATUS - HFCLK Status Register

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																														0x1		
Access																														R		
Name																														SELECTED		

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	SELECTED	0x1	R	**HFCLK Selected**

Clock selected as HFCLK clock source.

Value	Mode	Description
1	HFRCO	HFRCO is selected as HFCLK clock source
2	HFXO	HFXO is selected as HFCLK clock source
3	LFRCO	LFRCO is selected as HFCLK clock source
4	LFXO	LFXO is selected as HFCLK clock source
5	HFRCODIV2	HFRCO divided by 2 is selected as HFCLK clock source
6	USHFRCO	USHFRCO is selected as HFCLK clock source
7	CLKIN0	CLKIN0 is selected as HFCLK clock source

10.5.26 CMU_HFXOTRIMSTATUS - HFXO Trim Status

Offset	Bit Position																															
0x09C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0				0x000																0x000										
Access	R	R				R																R										
Name	MONVALID	VALID				IBTRIMXOCOREMON																IBTRIMXOCORE										

Bit	Name	Reset	Access	Description
31	MONVALID	0	R	**Peak Detection Algorithm or Peak Monitoring Algorithm Found a Value for IBTRIMXOCOREMON**
30	VALID	0	R	**Peak Detection Algorithm Found a Value for IBTRIMXOCORE** If HFXO is started again with PEAKDETTMODE=ONCECMD the IBTRIMXOCORE value from CMU_HFXOTRIMSTATUS will be used and peak detection algorithm will be skipped.
29:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:16	IBTRIMXOCOREMON	0x000	R	**Value of IBTRIMXOCORE Found By Automatic HFXO Peak Detection Algorithm or Peak Monitoring Algorithm (completion of Either Algorithm Will Cause an Update of IBTRIMXOCOREMON)**
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:0	IBTRIMXOCORE	0x000	R	**Value of IBTRIMXOCORE Found By Automatic HFXO Peak Detection Algorithm** If HFXO is started again with PEAKDETTMODE=ONCECMD this value will be used as steady state value for the HFXO (instead of the IBTRIMXOCORE value from CMU_HFXOSTEADYSTATECTRL) and peak detection algorithm will be skipped

10.5.27 CMU_IF - Interrupt Flag Register

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	0		0	0	0										0	0	0	0	0		0		0	0	0	0	0	0	0	0	0	1
Access	R		R	R	R										R	R	R	R	R		R		R	R	R	R	R	R	R	R	R	R
Name	CMUERR		ULFRCOEDGE	LFRCOEDGE	LFXOEDGE										DPLLLOCKFAILHIGH	DPLLLOCKFAILLOW	DPLLRDY	LFTIMEOUTERR	HFRCODIS		HFXOPEAKDETRDY		HFXOAUTOSW	HFXODISERR	USHFRCORDY	CALOF	CALRDY	AUXHFRCORDY	LFXORDY	LFRCORDY	HFXORDY	HFRCORDY

Bit	Name	Reset	Access	Description
31	CMUERR	0	R	**CMU Error Interrupt Flag**
	Set upon illegal CMU write attempt (e.g. writing CMU_LFRCOCTRL while LFRCOBSY is set).			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	ULFRCOEDGE	0	R	**ULFRCO Clock Edge Detected Interrupt Flag**
	Sets when ULFRCO clock switches phases.			
28	LFRCOEDGE	0	R	**LFRCO Clock Edge Detected Interrupt Flag**
	Sets when LFRCO clock switches phases.			
27	LFXOEDGE	0	R	**LFXO Clock Edge Detected Interrupt Flag**
	Sets when LFXO clock switches phases.			
26:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	DPLLLOCKFAILHIGH	0	R	**DPLL Lock Failure Low Interrupt Flag**
	Set when DPLL fail to lock because of period overflow.			
16	DPLLLOCKFAILLOW	0	R	**DPLL Lock Failure Low Interrupt Flag**
	Set when DPLL fail to lock because of period underflow.			
15	DPLLRDY	0	R	**DPLL Lock Interrupt Flag**
	Set when DPLL achieve the lock.			
14	LFTIMEOUTERR	0	R	**Low Frequency Timeout Error Interrupt Flag**
	Set when LFTIMEOUT of CMU_HFXOCTRL triggers before the combined STARTUPTIMEOUT plus STEADYTIMEOUT of the CMU_HFXOTIMEOUTCTRL register triggers.			
13	HFRCODIS	0	R	**HFRCO Disable Interrupt Flag**
	Set when a running HFRCO is disabled because of automatic HFXO start and selection.			
12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	HFXOPEAKDETRDY	0	R	**HFXO Automatic Peak Detection Ready Interrupt Flag**
	Set when automatic HFXO peak detection is ready.			

Bit	Name	Reset	Access	Description
10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	HFXOAUTOSW	0	R	**HFXO Automatic Switch Interrupt Flag**
	Set when automatic selection of HFXO causes a switch of the source clock used for HFCLKSRC.			
8	HFXODISERR	0	R	**HFXO Disable Error Interrupt Flag**
	Set when software tries to disable/deselect the HFXO in case the automatic enable/select reason is met. The HFXO was not disabled/deselected.			
7	USHFRCORDY	0	R	**USHFRCO Ready Interrupt Flag**
	Set when USHFRCO is ready (start-up time exceeded).			
6	CALOF	0	R	**Calibration Overflow Interrupt Flag**
	Set when calibration overflow has occurred (i.e. if a new calibration completes before CMU_CALCNT has been read).			
5	CALRDY	0	R	**Calibration Ready Interrupt Flag**
	Set when calibration is completed.			
4	AUXHFRCORDY	0	R	**AUXHFRCO Ready Interrupt Flag**
	Set when AUXHFRCO is ready (start-up time exceeded).			
3	LFXORDY	0	R	**LFXO Ready Interrupt Flag**
	Set when LFXO is ready (start-up time exceeded). LFXORDY can be used as wake-up interrupt.			
2	LFRCORDY	0	R	**LFRCO Ready Interrupt Flag**
	Set when LFRCO is ready (start-up time exceeded). LFRCORDY can be used as wake-up interrupt.			
1	HFXORDY	0	R	**HFXO Ready Interrupt Flag**
	Set when HFXO is ready (start-up time exceeded).			
0	HFRCORDY	1	R	**HFRCO Ready Interrupt Flag**
	Set when HFRCO is ready (start-up time exceeded).			

10.5.28 CMU_IFS - Interrupt Flag Set Register

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	0		0	0	0										0	0	0	0	0		0		0	0	0	0	0	0	0	0	0	0
Access	W1		W1	W1	W1										W1	W1	W1	W1	W1		W1		W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name	CMUERR		ULFRCOEDGE	LFRCOEDGE	LFXOEDGE										DPLLLOCKFAILHIGH	DPLLLOCKFAILLOW	DPLLRDY	LFTIMEOUTERR	HFRCODIS		HFXOPEAKDETRDY		HFXOAUTOSW	HFXODISERR	USHFRCORDY	CALOF	CALRDY	AUXHFRCORDY	LFXORDY	LFRCORDY	HFXORDY	HFRCORDY

Bit	Name	Reset	Access	Description
31	CMUERR	0	W1	**Set CMUERR Interrupt Flag**
	Write 1 to set the CMUERR interrupt flag			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	ULFRCOEDGE	0	W1	**Set ULFRCOEDGE Interrupt Flag**
	Write 1 to set the ULFRCOEDGE interrupt flag			
28	LFRCOEDGE	0	W1	**Set LFRCOEDGE Interrupt Flag**
	Write 1 to set the LFRCOEDGE interrupt flag			
27	LFXOEDGE	0	W1	**Set LFXOEDGE Interrupt Flag**
	Write 1 to set the LFXOEDGE interrupt flag			
26:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	DPLLLOCKFAILHIGH	0	W1	**Set DPLLLOCKFAILHIGH Interrupt Flag**
	Write 1 to set the DPLLLOCKFAILHIGH interrupt flag			
16	DPLLLOCKFAILLOW	0	W1	**Set DPLLLOCKFAILLOW Interrupt Flag**
	Write 1 to set the DPLLLOCKFAILLOW interrupt flag			
15	DPLLRDY	0	W1	**Set DPLLRDY Interrupt Flag**
	Write 1 to set the DPLLRDY interrupt flag			
14	LFTIMEOUTERR	0	W1	**Set LFTIMEOUTERR Interrupt Flag**
	Write 1 to set the LFTIMEOUTERR interrupt flag			
13	HFRCODIS	0	W1	**Set HFRCODIS Interrupt Flag**
	Write 1 to set the HFRCODIS interrupt flag			
12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	HFXOPEAKDETRDY	0	W1	**Set HFXOPEAKDETRDY Interrupt Flag**
	Write 1 to set the HFXOPEAKDETRDY interrupt flag			

Bit	Name	Reset	Access	Description
10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	HFXOAUTOSW	0	W1	**Set HFXOAUTOSW Interrupt Flag**
	Write 1 to set the HFXOAUTOSW interrupt flag			
8	HFXODISERR	0	W1	**Set HFXODISERR Interrupt Flag**
	Write 1 to set the HFXODISERR interrupt flag			
7	USHFRCORDY	0	W1	**Set USHFRCORDY Interrupt Flag**
	Write 1 to set the USHFRCORDY interrupt flag			
6	CALOF	0	W1	**Set CALOF Interrupt Flag**
	Write 1 to set the CALOF interrupt flag			
5	CALRDY	0	W1	**Set CALRDY Interrupt Flag**
	Write 1 to set the CALRDY interrupt flag			
4	AUXHFRCORDY	0	W1	**Set AUXHFRCORDY Interrupt Flag**
	Write 1 to set the AUXHFRCORDY interrupt flag			
3	LFXORDY	0	W1	**Set LFXORDY Interrupt Flag**
	Write 1 to set the LFXORDY interrupt flag			
2	LFRCORDY	0	W1	**Set LFRCORDY Interrupt Flag**
	Write 1 to set the LFRCORDY interrupt flag			
1	HFXORDY	0	W1	**Set HFXORDY Interrupt Flag**
	Write 1 to set the HFXORDY interrupt flag			
0	HFRCORDY	0	W1	**Set HFRCORDY Interrupt Flag**
	Write 1 to set the HFRCORDY interrupt flag			

10.5.29 CMU_IFC - Interrupt Flag Clear Register

Offset	Bit Position																													
0x0A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0		0	0	0										0	0	0	0	0		0		0	0	0	0	0	0	0	0	0	0
Access	(R)W1		(R)W1	(R)W1	(R)W1										(R)W1	(R)W1	(R)W1	(R)W1	(R)W1		(R)W1		(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name	CMUERR		ULFRCOEDGE	LFRCOEDGE	LFXOEDGE										DPLLLOCKFAILHIGH	DPLLLOCKFAILLOW	DPLLRDY	LFTIMEOUTERR	HFRCODIS		HFXOPEAKDETRDY		HFXOAUTOSW	HFXODISERR	USHFRCORDY	CALOF	CALRDY	AUXHFRCORDY	LFXORDY	LFRCORDY	HFXORDY	HFRCORDY

Bit	Name	Reset	Access	Description
31	CMUERR	0	(R)W1	**Clear CMUERR Interrupt Flag**
	Write 1 to clear the CMUERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	ULFRCOEDGE	0	(R)W1	**Clear ULFRCOEDGE Interrupt Flag**
	Write 1 to clear the ULFRCOEDGE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
28	LFRCOEDGE	0	(R)W1	**Clear LFRCOEDGE Interrupt Flag**
	Write 1 to clear the LFRCOEDGE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
27	LFXOEDGE	0	(R)W1	**Clear LFXOEDGE Interrupt Flag**
	Write 1 to clear the LFXOEDGE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
26:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	DPLLLOCKFAILHIGH	0	(R)W1	**Clear DPLLLOCKFAILHIGH Interrupt Flag**
	Write 1 to clear the DPLLLOCKFAILHIGH interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
16	DPLLLOCKFAILLOW	0	(R)W1	**Clear DPLLLOCKFAILLOW Interrupt Flag**
	Write 1 to clear the DPLLLOCKFAILLOW interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
15	DPLLRDY	0	(R)W1	**Clear DPLLRDY Interrupt Flag**
	Write 1 to clear the DPLLRDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
14	LFTIMEOUTERR	0	(R)W1	**Clear LFTIMEOUTERR Interrupt Flag**
	Write 1 to clear the LFTIMEOUTERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
13	HFRCODIS	0	(R)W1	**Clear HFRCODIS Interrupt Flag**
	Write 1 to clear the HFRCODIS interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	HFXOPEAKDETRDY	0	(R)W1	**Clear HFXOPEAKDETRDY Interrupt Flag**
	Write 1 to clear the HFXOPEAKDETRDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	HFXOAUTOSW	0	(R)W1	**Clear HFXOAUTOSW Interrupt Flag**
	Write 1 to clear the HFXOAUTOSW interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	HFXODISERR	0	(R)W1	**Clear HFXODISERR Interrupt Flag**
	Write 1 to clear the HFXODISERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	USHFRCORDY	0	(R)W1	**Clear USHFRCORDY Interrupt Flag**
	Write 1 to clear the USHFRCORDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
6	CALOF	0	(R)W1	**Clear CALOF Interrupt Flag**
	Write 1 to clear the CALOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
5	CALRDY	0	(R)W1	**Clear CALRDY Interrupt Flag**
	Write 1 to clear the CALRDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
4	AUXHFRCORDY	0	(R)W1	**Clear AUXHFRCORDY Interrupt Flag**
	Write 1 to clear the AUXHFRCORDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	LFXORDY	0	(R)W1	**Clear LFXORDY Interrupt Flag**
	Write 1 to clear the LFXORDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	LFRCORDY	0	(R)W1	**Clear LFRCORDY Interrupt Flag**
	Write 1 to clear the LFRCORDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	HFXORDY	0	(R)W1	**Clear HFXORDY Interrupt Flag**
	Write 1 to clear the HFXORDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	HFRCORDY	0	(R)W1	**Clear HFRCORDY Interrupt Flag**
	Write 1 to clear the HFRCORDY interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

10.5.30 CMU_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x0AC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0		0	0	0										0	0	0	0	0		0		0	0	0	0	0	0	0	0	0	0
Access	RW		RW	RW	RW										RW	RW	RW	RW	RW		RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name	CMUERR		ULFRCOEDGE	LFRCOEDGE	LFXOEDGE										DPLLLOCKFAILHIGH	DPLLLOCKFAILLOW	DPLLRDY	LFTIMEOUTERR	HFRCODIS		HFXOPEAKDETRDY		HFXOAUTOSW	HFXODISERR	USHFRCORDY	CALOF	CALRDY	AUXHFRCORDY	LFXORDY	LFRCORDY	HFXORDY	HFRCORDY

Bit	Name	Reset	Access	Description
31	CMUERR	0	RW	**CMUERR Interrupt Enable**
	Enable/disable the CMUERR interrupt			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	ULFRCOEDGE	0	RW	**ULFRCOEDGE Interrupt Enable**
	Enable/disable the ULFRCOEDGE interrupt			
28	LFRCOEDGE	0	RW	**LFRCOEDGE Interrupt Enable**
	Enable/disable the LFRCOEDGE interrupt			
27	LFXOEDGE	0	RW	**LFXOEDGE Interrupt Enable**
	Enable/disable the LFXOEDGE interrupt			
26:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	DPLLLOCKFAILHIGH	0	RW	**DPLLLOCKFAILHIGH Interrupt Enable**
	Enable/disable the DPLLLOCKFAILHIGH interrupt			
16	DPLLLOCKFAILLOW	0	RW	**DPLLLOCKFAILLOW Interrupt Enable**
	Enable/disable the DPLLLOCKFAILLOW interrupt			
15	DPLLRDY	0	RW	**DPLLRDY Interrupt Enable**
	Enable/disable the DPLLRDY interrupt			
14	LFTIMEOUTERR	0	RW	**LFTIMEOUTERR Interrupt Enable**
	Enable/disable the LFTIMEOUTERR interrupt			
13	HFRCODIS	0	RW	**HFRCODIS Interrupt Enable**
	Enable/disable the HFRCODIS interrupt			
12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	HFXOPEAKDETRDY	0	RW	**HFXOPEAKDETRDY Interrupt Enable**
	Enable/disable the HFXOPEAKDETRDY interrupt			

Bit	Name	Reset	Access	Description
10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	HFXOAUTOSW	0	RW	**HFXOAUTOSW Interrupt Enable**
	Enable/disable the HFXOAUTOSW interrupt			
8	HFXODISERR	0	RW	**HFXODISERR Interrupt Enable**
	Enable/disable the HFXODISERR interrupt			
7	USHFRCORDY	0	RW	**USHFRCORDY Interrupt Enable**
	Enable/disable the USHFRCORDY interrupt			
6	CALOF	0	RW	**CALOF Interrupt Enable**
	Enable/disable the CALOF interrupt			
5	CALRDY	0	RW	**CALRDY Interrupt Enable**
	Enable/disable the CALRDY interrupt			
4	AUXHFRCORDY	0	RW	**AUXHFRCORDY Interrupt Enable**
	Enable/disable the AUXHFRCORDY interrupt			
3	LFXORDY	0	RW	**LFXORDY Interrupt Enable**
	Enable/disable the LFXORDY interrupt			
2	LFRCORDY	0	RW	**LFRCORDY Interrupt Enable**
	Enable/disable the LFRCORDY interrupt			
1	HFXORDY	0	RW	**HFXORDY Interrupt Enable**
	Enable/disable the HFXORDY interrupt			
0	HFRCORDY	0	RW	**HFRCORDY Interrupt Enable**
	Enable/disable the HFRCORDY interrupt			

10.5.31 CMU_HFBUSCLKEN0 - High Frequency Bus Clock Enable Register 0

Offset	Bit Position																													
0x0B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0	0	0	0	0	0	0	0	0	0
Access																							RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																							USB	QSPI0	GPCRC	LDMA	PRS	GPIO	SDIO	EBI	CRYPTO0	LE

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	USB	0	RW	**Universal Serial Bus Interface Clock Enable**
	Set to enable the clock for USB.			
8	QSPI0	0	RW	**Quad-SPI Clock Enable**
	Set to enable the clock for QSPI0.			
7	GPCRC	0	RW	**General Purpose CRC Clock Enable**
	Set to enable the clock for GPCRC.			
6	LDMA	0	RW	**Linked Direct Memory Access Controller Clock Enable**
	Set to enable the clock for LDMA.			
5	PRS	0	RW	**Peripheral Reflex System Clock Enable**
	Set to enable the clock for PRS.			
4	GPIO	0	RW	**General purpose Input/Output Clock Enable**
	Set to enable the clock for GPIO.			
3	SDIO	0	RW	**SDIO Controller Clock Enable**
	Set to enable the clock for SDIO.			
2	EBI	0	RW	**External Bus Interface Clock Enable**
	Set to enable the clock for EBI.			
1	CRYPTO0	0	RW	**Advanced Encryption Standard Accelerator Clock Enable**
	Set to enable the clock for CRYPTO0.			
0	LE	0	RW	**Low Energy Peripheral Interface Clock Enable**
	Set to enable the clock for LE. Interface used for bus access to Low Energy peripherals.			

10.5.32 CMU_HFPERCLKEN0 - High Frequency Peripheral Clock Enable Register 0

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													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access													RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name													TRNG0	IDAC0	CRYOTIMER	PDM	ADC1	ADC0	I2C1	I2C0	ACMP2	ACMP1	ACMP0	TIMER3	TIMER2	TIMER1	TIMER0	USART4	USART3	USART2	USART1	USART0

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	TRNG0	0	RW	**True Random Number Generator 0 Clock Enable**
	Set to enable the clock for TRNG0.			
18	IDAC0	0	RW	**Current Digital to Analog Converter 0 Clock Enable**
	Set to enable the clock for IDAC0.			
17	CRYOTIMER	0	RW	**CRYOTIMER Clock Enable**
	Set to enable the clock for CRYOTIMER.			
16	PDM	0	RW	**PDM Interface Clock Enable**
	Set to enable the clock for PDM.			
15	ADC1	0	RW	**Analog to Digital Converter 0 Clock Enable**
	Set to enable the clock for ADC1.			
14	ADC0	0	RW	**Analog to Digital Converter 0 Clock Enable**
	Set to enable the clock for ADC0.			
13	I2C1	0	RW	**I2C 1 Clock Enable**
	Set to enable the clock for I2C1.			
12	I2C0	0	RW	**I2C 0 Clock Enable**
	Set to enable the clock for I2C0.			
11	ACMP2	0	RW	**Analog Comparator 2 Clock Enable**
	Set to enable the clock for ACMP2.			
10	ACMP1	0	RW	**Analog Comparator 1 Clock Enable**
	Set to enable the clock for ACMP1.			
9	ACMP0	0	RW	**Analog Comparator 0 Clock Enable**
	Set to enable the clock for ACMP0.			
8	TIMER3	0	RW	**Timer 3 Clock Enable**
	Set to enable the clock for TIMER3.			
7	TIMER2	0	RW	**Timer 2 Clock Enable**
	Set to enable the clock for TIMER2.			

Bit	Name	Reset	Access	Description
6	TIMER1	0	RW	**Timer 1 Clock Enable**
	Set to enable the clock for TIMER1.			
5	TIMER0	0	RW	**Timer 0 Clock Enable**
	Set to enable the clock for TIMER0.			
4	USART4	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 4 Clock Enable**
	Set to enable the clock for USART4.			
3	USART3	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 3 Clock Enable**
	Set to enable the clock for USART3.			
2	USART2	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 2 Clock Enable**
	Set to enable the clock for USART2.			
1	USART1	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 1 Clock Enable**
	Set to enable the clock for USART1.			
0	USART0	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 0 Clock Enable**
	Set to enable the clock for USART0.			

10.5.33 CMU_HFPERCLKEN1 - High Frequency Peripheral Clock Enable Register 1

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																									0	0	0	0	0	0	0	0
Access																									RW	RW	RW	RW	RW	RW	RW	RW
Name																									CSEN	VDAC0	CAN1	CAN0	WTIMER1	WTIMER0	UART1	UART0

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	CSEN	0	RW	**Capacitive touch sense module Clock Enable**
	Set to enable the clock for CSEN.			
6	VDAC0	0	RW	**Digital to Analog Converter 0 Clock Enable**
	Set to enable the clock for VDAC0.			
5	CAN1	0	RW	**CAN 1 Clock Enable**
	Set to enable the clock for CAN1.			
4	CAN0	0	RW	**CAN 0 Clock Enable**
	Set to enable the clock for CAN0.			
3	WTIMER1	0	RW	**Wide Timer 0 Clock Enable**
	Set to enable the clock for WTIMER1.			
2	WTIMER0	0	RW	**Wide Timer 0 Clock Enable**
	Set to enable the clock for WTIMER0.			
1	UART1	0	RW	**Universal Asynchronous Receiver/Transmitter 1 Clock Enable**
	Set to enable the clock for UART1.			
0	UART0	0	RW	**Universal Asynchronous Receiver/Transmitter 0 Clock Enable**
	Set to enable the clock for UART0.			

10.5.34 CMU_LFACLKEN0 - Low Frequency a Clock Enable Register 0 (Async Reg)

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																												0	0	0	0	0
Access																												RW	RW	RW	RW	RW
Name																												RTC	LCD	LESENSE	LETIMER1	LETIMER0

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	RTC	0	RW	**Real-Time Counter Clock Enable**
	Set to enable the clock for RTC.			
3	LCD	0	RW	**Liquid Crystal Display Controller Clock Enable**
	Set to enable the clock for LCD.			
2	LESENSE	0	RW	**Low Energy Sensor Interface Clock Enable**
	Set to enable the clock for LESENSE.			
1	LETIMER1	0	RW	**Low Energy Timer 1 Clock Enable**
	Set to enable the clock for LETIMER1.			
0	LETIMER0	0	RW	**Low Energy Timer 0 Clock Enable**
	Set to enable the clock for LETIMER0.			

10.5.35 CMU_LFBCLKEN0 - Low Frequency B Clock Enable Register 0 (Async Reg)

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																													0	0	0	0
Access																													RW	RW	RW	RW
Name																													CSEN	SYSTICK	LEUART1	LEUART0

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	CSEN	0	RW	**Capacitive touch sense module Clock Enable**
	Set to enable the clock for CSEN.			
2	SYSTICK	0	RW	**Clock Enable**
	Set to enable the clock for SYSTICK.			
1	LEUART1	0	RW	**Low Energy UART 1 Clock Enable**
	Set to enable the clock for LEUART1.			
0	LEUART0	0	RW	**Low Energy UART 0 Clock Enable**
	Set to enable the clock for LEUART0.			

10.5.36 CMU_LFCCLKEN0 - Low Frequency C Clock Enable Register 0 (Async Reg)

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																																0
Access																																RW
Name																																USB

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	USB	0	RW	**Universal Serial Bus Interface Clock Enable**
	Set to enable the clock for USB.			

10.5.37 CMU_LFECLKEN0 - Low Frequency E Clock Enable Register 0 (Async Reg)

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																																0
Access																																RW
Name																																RTCC

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	RTCC	0	RW	**Real-Time Counter and Calendar Clock Enable**
	Set to enable the clock for RTCC.			

10.5.38 CMU_HFPRESC - High Frequency Clock Prescaler 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													0x00											
Access							RW													RW											
Name							HFCLKLEPRESC													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	HFCLKLEPRESC	0x0	RW	**HFCLKLE Prescaler**

Specifies the clock divider for HFCLKLE.

Value	Mode	Description
0	DIV2	HFCLKLE is $HFBUSCLK_{LE}$ divided by 2.
1	DIV4	HFCLKLE is $HFBUSCLK_{LE}$ divided by 4.
2	DIV8	HFCLKLE is $HFBUSCLK_{LE}$ divided by 8.

23:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:8	PRESC	0x00	RW	**HFCLK Prescaler**

Specifies the clock divider for HFCLK (relative to HFSRCCLK).

Value	Description
PRESC	Clock division factor of PRESC+1.

7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.39 CMU_HFBUSPRESC - High Frequency Bus Clock Prescaler 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																				0x000												
Access																				RW												
Name																				PRESC												

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*		
16:8	PRESC	0x000	RW	**HFBUSCLK Prescaler**
	Specifies the clock divider for the HFBUSCLK (relative to HFCLK).			
	Value			Description
	PRESC			Clock division factor of PRESC+1.
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.40 CMU_HFCOREPRESC - High Frequency Core Clock Prescaler Register

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																				0x000												
Access																				RW												
Name																				PRESC												

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*		
16:8	PRESC	0x000	RW	**HFCORECLK Prescaler**
	Specifies the clock divider for HFCORECLK (relative to HFCLK).			
	Value			Description
	PRESC			Clock division factor of PRESC+1.
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.41 CMU_HFPERPRESC - High Frequency Peripheral Clock Prescaler Register

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																				0x000												
Access																				RW												
Name																				PRESC												

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*		
16:8	PRESC	0x000	RW	**HFPERCLK Prescaler**

Specifies the clock divider for the HFPERCLK (relative to HFCLK).

Value	Description
PRESC	Clock division factor of PRESC+1.

Bit	Name	Reset	Access	Description
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.42 CMU_HFEXPPRESC - High Frequency Export Clock Prescaler Register

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																						0x00										
Access																						RW										
Name																						PRESC										

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*		
12:8	PRESC	0x00	RW	**HFEXPCLK Prescaler**

Specifies the clock divider for HFEXPCLK (relative to HFCLK).

Value	Description
PRESC	Clock division factor of PRESC+1.

Bit	Name	Reset	Access	Description
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.43 CMU_HFPERPRESCB - High Frequency Peripheral Clock Prescaler B Register

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																				0x000												
Access																				RW												
Name																				PRESC												

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*		
16:8	PRESC	0x000	RW	**HFPERCLK Prescaler**

Specifies the clock divider for the HFPERCLK (relative to HFCLK).

Value	Description
PRESC	Clock division factor of PRESC+1.

Bit	Name	Reset	Access	Description
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.44 CMU_HFPERPRESCC - High Frequency Peripheral Clock Prescaler C Register

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																				0x000												
Access																				RW												
Name																				PRESC												

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*		
16:8	PRESC	0x000	RW	**HFPERCLK Prescaler**

Specifies the clock divider for the HFPERCLK (relative to HFCLK).

Value	Description
PRESC	Clock division factor of PRESC+1.

Bit	Name	Reset	Access	Description
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

10.5.45 CMU_LFAPRESC0 - Low Frequency a Prescaler Register 0 (Async Reg)

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					0x0					0x0		0x0				0x0			
Access													RW					RW					RW		RW				RW			
Name													RTC					LCD					LESENSE		LETIMER1				LETIMER0			

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	RTC	0x0	RW	**Real-Time Counter Prescaler**

Configure Real-Time Counter prescaler

Value	Mode	Description
0	DIV1	$LFACLK_{RTC}$ = LFACLK
1	DIV2	$LFACLK_{RTC}$ = LFACLK/2
2	DIV4	$LFACLK_{RTC}$ = LFACLK/4
3	DIV8	$LFACLK_{RTC}$ = LFACLK/8
4	DIV16	$LFACLK_{RTC}$ = LFACLK/16
5	DIV32	$LFACLK_{RTC}$ = LFACLK/32
6	DIV64	$LFACLK_{RTC}$ = LFACLK/64
7	DIV128	$LFACLK_{RTC}$ = LFACLK/128
8	DIV256	$LFACLK_{RTC}$ = LFACLK/256
9	DIV512	$LFACLK_{RTC}$ = LFACLK/512
10	DIV1024	$LFACLK_{RTC}$ = LFACLK/1024
11	DIV2048	$LFACLK_{RTC}$ = LFACLK/2048
12	DIV4096	$LFACLK_{RTC}$ = LFACLK/4096
13	DIV8192	$LFACLK_{RTC}$ = LFACLK/8192
14	DIV16384	$LFACLK_{RTC}$ = LFACLK/16384
15	DIV32768	$LFACLK_{RTC}$ = LFACLK/32768

15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:12	LCD	0x0	RW	**Liquid Crystal Display Controller Prescaler**

Configure Liquid Crystal Display Controller prescaler

Value	Mode	Description
0	DIV1	$LFACLK_{LCD}$ = LFACLK

Bit	Name	Reset	Access	Description
	1	DIV2		$LFACLK_{LCD}$ = LFACLK/2
	2	DIV4		$LFACLK_{LCD}$ = LFACLK/4
	3	DIV8		$LFACLK_{LCD}$ = LFACLK/8
	4	DIV16		$LFACLK_{LCD}$ = LFACLK/16
	5	DIV32		$LFACLK_{LCD}$ = LFACLK/32
	6	DIV64		$LFACLK_{LCD}$ = LFACLK/64
	7	DIV128		$LFACLK_{LCD}$ = LFACLK/128
11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	LESENSE	0x0	RW	**Low Energy Sensor Interface Prescaler**
	Configure Low Energy Sensor Interface prescaler			
	Value	Mode		Description
	0	DIV1		$LFACLK_{LESENSE}$ = LFACLK
	1	DIV2		$LFACLK_{LESENSE}$ = LFACLK/2
	2	DIV4		$LFACLK_{LESENSE}$ = LFACLK/4
	3	DIV8		$LFACLK_{LESENSE}$ = LFACLK/8
7:4	LETIMER1	0x0	RW	**Low Energy Timer 1 Prescaler**
	Configure Low Energy Timer 1 prescaler			
	Value	Mode		Description
	0	DIV1		$LFACLK_{LETIMER1}$ = LFACLK
	1	DIV2		$LFACLK_{LETIMER1}$ = LFACLK/2
	2	DIV4		$LFACLK_{LETIMER1}$ = LFACLK/4
	3	DIV8		$LFACLK_{LETIMER1}$ = LFACLK/8
	4	DIV16		$LFACLK_{LETIMER1}$ = LFACLK/16
	5	DIV32		$LFACLK_{LETIMER1}$ = LFACLK/32
	6	DIV64		$LFACLK_{LETIMER1}$ = LFACLK/64
	7	DIV128		$LFACLK_{LETIMER1}$ = LFACLK/128
	8	DIV256		$LFACLK_{LETIMER1}$ = LFACLK/256
	9	DIV512		$LFACLK_{LETIMER1}$ = LFACLK/512
	10	DIV1024		$LFACLK_{LETIMER1}$ = LFACLK/1024
	11	DIV2048		$LFACLK_{LETIMER1}$ = LFACLK/2048
	12	DIV4096		$LFACLK_{LETIMER1}$ = LFACLK/4096
	13	DIV8192		$LFACLK_{LETIMER1}$ = LFACLK/8192
	14	DIV16384		$LFACLK_{LETIMER1}$ = LFACLK/16384

Bit	Name	Reset	Access	Description
	15	DIV32768		$LFACLK_{LETIMER1}$ = LFACLK/32768
3:0	LETIMER0	0x0	RW	**Low Energy Timer 0 Prescaler**
	Configure Low Energy Timer 0 prescaler			

Value	Mode	Description
0	DIV1	$LFACLK_{LETIMER0}$ = LFACLK
1	DIV2	$LFACLK_{LETIMER0}$ = LFACLK/2
2	DIV4	$LFACLK_{LETIMER0}$ = LFACLK/4
3	DIV8	$LFACLK_{LETIMER0}$ = LFACLK/8
4	DIV16	$LFACLK_{LETIMER0}$ = LFACLK/16
5	DIV32	$LFACLK_{LETIMER0}$ = LFACLK/32
6	DIV64	$LFACLK_{LETIMER0}$ = LFACLK/64
7	DIV128	$LFACLK_{LETIMER0}$ = LFACLK/128
8	DIV256	$LFACLK_{LETIMER0}$ = LFACLK/256
9	DIV512	$LFACLK_{LETIMER0}$ = LFACLK/512
10	DIV1024	$LFACLK_{LETIMER0}$ = LFACLK/1024
11	DIV2048	$LFACLK_{LETIMER0}$ = LFACLK/2048
12	DIV4096	$LFACLK_{LETIMER0}$ = LFACLK/4096
13	DIV8192	$LFACLK_{LETIMER0}$ = LFACLK/8192
14	DIV16384	$LFACLK_{LETIMER0}$ = LFACLK/16384
15	DIV32768	$LFACLK_{LETIMER0}$ = LFACLK/32768

10.5.46 CMU_LFBPRESC0 - Low Frequency B Prescaler Register 0 (Async Reg)

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						0x0				0x0	
Access																			RW								RW				RW	
Name																			CSEN		SYSTICK						LEUART1				LEUART0	

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	CSEN	0x0	RW	**Capacitive touch sense module Prescaler**

Configure Capacitive touch sense module prescaler

Value	Mode	Description
0	DIV16	$LFBCLK_{CSEN}$ = LFBCLK/16
1	DIV32	$LFBCLK_{CSEN}$ = LFBCLK/32
2	DIV64	$LFBCLK_{CSEN}$ = LFBCLK/64
3	DIV128	$LFBCLK_{CSEN}$ = LFBCLK/128

Bit	Name	Reset	Access	Description
11:8	SYSTICK	0x0		**Prescaler**

Configure prescaler

Value	Mode	Description
0	DIV1	$LFBCLK_{SYSTICK}$ = LFBCLK

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	LEUART1	0x0	RW	**Low Energy UART 1 Prescaler**

Configure Low Energy UART 1 prescaler

Value	Mode	Description
0	DIV1	$LFBCLK_{LEUART1}$ = LFBCLK
1	DIV2	$LFBCLK_{LEUART1}$ = LFBCLK/2
2	DIV4	$LFBCLK_{LEUART1}$ = LFBCLK/4
3	DIV8	$LFBCLK_{LEUART1}$ = LFBCLK/8

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	LEUART0	0x0	RW	**Low Energy UART 0 Prescaler**

Configure Low Energy UART 0 prescaler

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	DIV1		$LFBCLK_{LEUART0}$ = LFBCLK
	1	DIV2		$LFBCLK_{LEUART0}$ = LFBCLK/2
	2	DIV4		$LFBCLK_{LEUART0}$ = LFBCLK/4
	3	DIV8		$LFBCLK_{LEUART0}$ = LFBCLK/8

10.5.47 CMU_LFEPRESC0 - Low Frequency E Prescaler Register 0 (Async Reg)

When waking up from EM4 make sure EM4UNLATCH in EMU_CMD is set for this to take effect

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																															RTCC	

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	RTCC	0x0	RW	**Real-Time Counter and Calendar Prescaler**

Configure Real-Time Counter and Calendar prescaler

Value	Mode	Description
0	DIV1	$LFECLK_{RTCC}$ = LFECLK
1	DIV2	$LFECLK_{RTCC}$ = LFECLK/2
2	DIV4	$LFECLK_{RTCC}$ = LFECLK/4

10.5.48 CMU_SYNCBUSY - Synchronization Busy Register

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		0	0	0	0	0	0	0						0		0								0		0		0		0		0
Access		R	R	R	R	R	R	R						R		R								R		R		R		R		R
Name		USHFRCOBSY	LFXOBSY	HFXOBSY	LFRCOVREFBSY	LFRCOBSY	AUXHFRCOBSY	HFRCOBSY						LFEPRESC0		LFECLKEN0								LFCCLKEN0		LFBPRESC0		LFBCLKEN0		LFAPRESC0		LFACLKEN0

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	USHFRCOBSY	0	R	**USHFRCO Busy**

Used to check the synchronization status of CMU_USHFRCOCTRL.

Value	Description
0	CMU_USHFRCOCTRL is ready for update
1	CMU_USHFRCOCTRL is busy synchronizing new value

29	LFXOBSY	0	R	**LFXO Busy**

Used to check the synchronization status of CMU_LFXOCTRL.

Value	Description
0	CMU_LFXOCTRL is ready for update
1	CMU_LFXOCTRL is busy synchronizing new value

28	HFXOBSY	0	R	**HFXO Busy**

Used to check the synchronization status of CMU_HFXOCTRL, CMU_HFXOSTARTUPCTRL, CMU_HFXOSTEADYSTATECTRL, CMU_HFXOTIMEOUTCTRL, CMU_HFXOCTRL1.

Value	Description
0	CMU_HFXOCTRL, CMU_HFXOSTARTUPCTRL, CMU_HFXOSTEADYSTATECTRL, CMU_HFXOTIMEOUTCTRL, CMU_HFXOCTRL1 are ready for update
1	CMU_HFXOCTRL, CMU_HFXOSTARTUPCTRL, CMU_HFXOSTEADYSTATECTRL, CMU_HFXOTIMEOUTCTRL, CMU_HFXOCTRL1 are busy synchronizing new value. HFXO is also BUSY when these registers are actively being used (e.g. when HFXOENS=1).

27	LFRCOVREFBSY	0	R	**LFRCO VREF Busy**

Used to check the synchronization status of GMCCURTUNE.

Value	Description
0	CMU_LFRCOCTRL GMCCURTUNE bitfield is ready for update

Bit	Name	Reset	Access	Description
	1			CMU_LFRCOCTRL GMCCURTUNE bitfield is busy synchronizing new value
26	LFRCOBSY	0	R	**LFRCO Busy**
	Used to check the synchronization status of CMU_LFRCOCTRL.			
	Value			Description
	0			CMU_LFRCOCTRL is ready for update
	1			CMU_LFRCOCTRL is busy synchronizing new value
25	AUXHFRCOBSY	0	R	**AUXHFRCO Busy**
	Used to check the synchronization status of CMU_AUXHFRCOCTRL.			
	Value			Description
	0			CMU_AUXHFRCOCTRL is ready for update
	1			CMU_AUXHFRCOCTRL is busy synchronizing new value
24	HFRCOBSY	0	R	**HFRCO Busy**
	Used to check the synchronization status of CMU_HFRCOCTRL.			
	Value			Description
	0			CMU_HFRCOCTRL is ready for update
	1			CMU_HFRCOCTRL is busy synchronizing new value
23:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	LFEPRESC0	0	R	**Low Frequency E Prescaler 0 Busy**
	Used to check the synchronization status of CMU_LFEPRESC0.			
	Value			Description
	0			CMU_LFEPRESC0 is ready for update
	1			CMU_LFEPRESC0 is busy synchronizing new value
17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	LFECLKEN0	0	R	**Low Frequency E Clock Enable 0 Busy**
	Used to check the synchronization status of CMU_LFECLKEN0.			
	Value			Description
	0			CMU_LFECLKEN0 is ready for update
	1			CMU_LFECLKEN0 is busy synchronizing new value
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	LFCCLKEN0	0	R	**Low Frequency C Clock Enable 0 Busy**
	Used to check the synchronization status of CMU_LFCCLKEN0.			

Bit	Name	Reset	Access	Description
	Value			Description
	0			CMU_LFCCLKEN0 is ready for update.
	1			CMU_LFCCLKEN0 is busy synchronizing new value.
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	LFBPRESC0	0	R	**Low Frequency B Prescaler 0 Busy**
	Used to check the synchronization status of CMU_LFBPRESC0.			
	Value			Description
	0			CMU_LFBPRESC0 is ready for update
	1			CMU_LFBPRESC0 is busy synchronizing new value
5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	LFBCLKEN0	0	R	**Low Frequency B Clock Enable 0 Busy**
	Used to check the synchronization status of CMU_LFBCLKEN0.			
	Value			Description
	0			CMU_LFBCLKEN0 is ready for update
	1			CMU_LFBCLKEN0 is busy synchronizing new value
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	LFAPRESC0	0	R	**Low Frequency a Prescaler 0 Busy**
	Used to check the synchronization status of CMU_LFAPRESC0.			
	Value			Description
	0			CMU_LFAPRESC0 is ready for update
	1			CMU_LFAPRESC0 is busy synchronizing new value
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	LFACLKEN0	0	R	**Low Frequency a Clock Enable 0 Busy**
	Used to check the synchronization status of CMU_LFACLKEN0.			
	Value			Description
	0			CMU_LFACLKEN0 is ready for update
	1			CMU_LFACLKEN0 is busy synchronizing new value

10.5.49 CMU_FREEZE - Freeze Register

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																																0
Access																																RW
Name																																REGFREEZE

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	REGFREEZE	0	RW	**Register Update Freeze**

When set, the update of the Low Frequency clock control registers is postponed until this bit is cleared. Use this bit to update several registers simultaneously.

Value	Mode	Description
0	UPDATE	Each write access to a Low Frequency clock control register is updated into the Low Frequency domain as soon as possible.
1	FREEZE	The LE Clock Control registers are not updated with the new written value.

10.5.50 CMU_PCNTCTRL - PCNT Control Register

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																											0	0	0	0	0	0
Access																											RWH	RWH	RWH	RWH	RWH	RWH
Name																											PCNT2CLKSEL	PCNT2CLKEN	PCNT1CLKSEL	PCNT1CLKEN	PCNT0CLKSEL	PCNT0CLKEN

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	PCNT2CLKSEL	0	RWH	**PCNT2 Clock Select**

This bit controls which clock that is used for the PCNT.

Value	Mode	Description
0	LFACLK	LFACLK is clocking PCNT2
1	PCNT2S0	External pin PCNT2_S0 is clocking PCNT0

Bit	Name	Reset	Access	Description
4	PCNT2CLKEN	0	RWH	**PCNT2 Clock Enable**

This bit enables/disables the clock to the PCNT.

Value	Description
0	PCNT2 is disabled
1	PCNT2 is enabled

Bit	Name	Reset	Access	Description
3	PCNT1CLKSEL	0	RWH	**PCNT1 Clock Select**

This bit controls which clock that is used for the PCNT.

Value	Mode	Description
0	LFACLK	LFACLK is clocking PCNT0
1	PCNT1S0	External pin PCNT1_S0 is clocking PCNT0

Bit	Name	Reset	Access	Description
2	PCNT1CLKEN	0	RWH	**PCNT1 Clock Enable**

This bit enables/disables the clock to the PCNT.

Value	Description
0	PCNT1 is disabled
1	PCNT1 is enabled

Bit	Name	Reset	Access	Description
1	PCNT0CLKSEL	0	RWH	**PCNT0 Clock Select**

This bit controls which clock that is used for the PCNT.

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	LFACLK		LFACLK is clocking PCNT0
	1	PCNT0S0		External pin PCNT0_S0 is clocking PCNT0
0	PCNT0CLKEN	0	RWH	**PCNT0 Clock Enable**
	This bit enables/disables the clock to the PCNT.			

10.5.51 CMU_ADCCTRL - ADC Control Register

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								0			0x0				0x0									0			0x0				0x0	
Access								RWH			RWH				RWH									RWH			RWH				RWH	
Name								ADC1CLKINV			ADC1CLKSEL				ADC1CLKDIV									ADC0CLKINV			ADC0CLKSEL				ADC0CLKDIV	

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	ADC1CLKINV	0	RWH	**Invert Clock Selected By ADC1CLKSEL**
	This bit enables inverting the selected clock to ADC1.			
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	ADC1CLKSEL	0x0	RWH	**ADC1 Clock Select**

This bit controls which clock is used for ADC1 in case ADCCLKMODE in ADCn_CTRL is set to ASYNC. It should only be changed when ADCCLKMODE in ADCn_CTRL is set to SYNC. HFXO should never be selected as clock source for ADC1 when disabling the HFXO (e.g. because of EM2 entry).

Value	Mode	Description
0	DISABLED	ADC1 is not clocked
1	AUXHFRCO	AUXHFRCO is clocking ADC1
2	HFXO	HFXO is clocking ADC1
3	HFSRCCLK	HFSRCCLK is clocking ADC1

19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	ADC1CLKDIV	0x0	RWH	**ADC1 Clock Prescaler**

Specifies the clock divider for ADC1.

Value	Description
PRESC	Clock division factor of PRESC+1.

15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	ADC0CLKINV	0	RWH	**Invert Clock Selected By ADC0CLKSEL**
	This bit enables inverting the selected clock to ADC0.			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
5:4	ADC0CLKSEL	0x0	RWH	**ADC0 Clock Select**

This bit controls which clock is used for ADC0 in case ADCCLKMODE in ADCn_CTRL is set to ASYNC. It should only be changed when ADCCLKMODE in ADCn_CTRL is set to SYNC. HFXO should never be selected as clock source for ADC0 when disabling the HFXO (e.g. because of EM2 entry).

Value	Mode	Description
0	DISABLED	ADC0 is not clocked
1	AUXHFRCO	AUXHFRCO is clocking ADC0
2	HFXO	HFXO is clocking ADC0
3	HFSRCCLK	HFSRCCLK is clocking ADC0

3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	ADC0CLKDIV	0x0	RWH	**ADC0 Clock Prescaler**

Specifies the clock divider for ADC0.

Value	Description
PRESC	Clock division factor of PRESC+1.

10.5.52 CMU_SDIOCTRL - SDIO Control Register

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																									0						0x0	
Access																									RW						RW	
Name																									SDIOCLKDIS						SDIOCLKSEL	

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	SDIOCLKDIS	0	RW	**SDIO Reference Clock Disable**
	Disables SDIO reference clock.			
6:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	SDIOCLKSEL	0x0	RW	**SDIO Reference Clock Select**
	Controls which clock is used for the SDIO reference clock.			

Value	Mode	Description
0	HFRCO	HFRCO clock is used to clock SDIO
1	HFXO	HFXO clock is used to clock SDIO
2	AUXHFRCO	AUXHFRCO is used to clock SDIO
3	USHFRCO	USHFRCO is used to clock SDIO

10.5.53 CMU_QSPICTRL - QSPI Control Register

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																									0						0x0	
Access																									RW						RW	
Name																									QSPI0CLKDIS						QSPI0CLKSEL	

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	QSPI0CLKDIS	0	RW	**QSPI0 Reference Clock Disable**
	Disables QSPI0 reference clock.			
6:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	QSPI0CLKSEL	0x0	RW	**QSPI0 Reference Clock Select**
	Controls which clock is used for the QSPI0 reference clock.			

Value	Mode	Description
0	HFRCO	HFRCO clock is used to clock QSPI0
1	HFXO	HFXO clock is used to clock QSPI0
2	AUXHFRCO	AUXHFRCO is used to clock QSPI0
3	USHFRCO	USHFRCO is used to clock QSPI0

10.5.54 CMU_PDMCTRL - PDM Control Register

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																									0						0x0	
Access																									RW						RW	
Name																									PDMCLKEN						PDMCLKSEL	

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	PDMCLKEN	0	RW	**PDM Core Clock Enable**
	Enables PDM Core clock.			
6:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PDMCLKSEL	0x0	RW	**PDM Core Clock Select**
	Controls which clock is used for the PDM reference clock.			

Value	Mode	Description
0	HFRCO	HFRCO clock is used to clock PDM
1	HFXO	HFXO clock is used to clock PDM
2	USHFRCO	USHFRCO is used to clock PDM
3	CLKIN0	CLKIN0 is selected as HFCLK clock source

10.5.55 CMU_ROUTEPEN - I/O Routing Pin Enable Register

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				0																										0	0	0
Access				RW																										RW	RW	RW
Name				CLKIN0PEN																										CLKOUT2PEN	CLKOUT1PEN	CLKOUT0PEN

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	CLKIN0PEN	0	RW	**CLKIN0 Pin Enable**
	When set, the CLKIN0 pin is enabled.			
27:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	CLKOUT2PEN	0	RW	**CLKOUT2 Pin Enable**
	When set, the CLKOUT2 pin is enabled.			
1	CLKOUT1PEN	0	RW	**CLKOUT1 Pin Enable**
	When set, the CLKOUT1 pin is enabled.			
0	CLKOUT0PEN	0	RW	**CLKOUT0 Pin Enable**
	When set, the CLKOUT0 pin is enabled.			

10.5.56 CMU_ROUTELOC0 - I/O Routing Location Register

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											0x00								0x00								0x00					
Access											RW								RW								RW					
Name											CLKOUT2LOC								CLKOUT1LOC								CLKOUT0LOC					

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	CLKOUT2LOC	0x00	RW	**I/O Location**

Decides the location of the CLKOUT2.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CLKOUT1LOC	0x00	RW	**I/O Location**

Decides the location of the CLKOUT1.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	CLKOUT0LOC	0x00	RW	**I/O Location**

Decides the location of the CMU CLKOUT0.

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2
	3	LOC3		Location 3
	4	LOC4		Location 4
	5	LOC5		Location 5

10.5.57 CMU_ROUTELOC1 - I/O Routing Location Register

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																											0x00					
Access																											RW					
Name																											CLKIN0LOC					

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	CLKIN0LOC	0x00	RW	**I/O Location**

Decides the location of the CLKIN0.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

10.5.58 CMU_LOCK - Configuration Lock Register

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																	0x0000															
Access																	RWH															
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	0x0000	RWH	**Configuration Lock Key**

Write any other value than the unlock code to lock CMU_CTRL, CMU_ROUTEPEN, CMU_ROUTELOC0, CMU_ROUTELOC1, CMU_HFRCOCTRL, CMU_AUXHFRCOCTRL, CMU_USHFRCOCTRL, CMU_LFRCOCTRL, CMU_ULFRCOCTRL, CMU_HFXOCTRL, CMU_HFXOCTRL1, CMU_HFXOSTARTUPCTRL, CMU_HFXOSTEADYSTATECTRL, CMU_HFXOTIMEOUTCTRL, CMU_LFXOCTRL, CMU_OSCENCMD, CMU_CMD, CMU_DBGCLKSEL, CMU_HFCLKSEL, CMU_LFACLKSEL, CMU_LFBCLKSEL, CMU_LFCCLKSEL, CMU_LFECLKSEL, CMU_HFBUSCLKEN0, CMU_HFPERCLKEN0, CMU_HFPERCLKEN1, CMU_HFPRESC, CMU_HFBUSPRESC, CMU_HFCOREPRESC, CMU_HFPERPRESC, CMU_HFPERPRESCB, CMU_HFPERPRESCC, CMU_HFEXPPRESC, CMU_LFACLKEN0, CMU_LFBCLKEN0, CMU_LFCCLKEN0, CMU_LFECLKEN0, CMU_LFAPRESC0, CMU_LFBPRESC0, CMU_LFEPRESC0, CMU_ADCCTRL CMU_SDIOCTRL, CMU_QSPICTRL, CMU_USBCTRL, and CMU_PCNTCTRL from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.

Mode	Value	Description
Read Operation		
UNLOCKED	0	CMU registers are unlocked
LOCKED	1	CMU registers are locked
Write Operation		
LOCK	0	Lock CMU registers
UNLOCK	0x580E	Unlock CMU registers

10.5.59 CMU_HFRCOSS - HFRCO Spread Spectrum Register

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																				0x00										0x0		
Access																				RW										RW		
Name																				SSINV										SSAMP		

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*		
12:8	SSINV	0x00	RW	**Spread Spectrum Update Interval** This value sets the update rate of the DCO period for spectrum spreading.
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	SSAMP	0x0	RW	**Spread Spectrum Amplitude** Randomize DCO output period with a peak-to-peak amplitude. Clear SSAMP would disable spectrum spreading.

10.5.60 CMU_USBCTRL - USB Control Register

Offset	Bit Position																															
0x1F0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0					0x0		
Access																									RW					RWH		
Name																									USBCLKEN					USBCLKSEL		

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	USBCLKEN	0	RW	**USB Rate Clock Enable**
	Enables USB rate clock.			
6:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	USBCLKSEL	0x0	RWH	**USB Rate Clock Select**
	Controls which clock is used for the USB.			

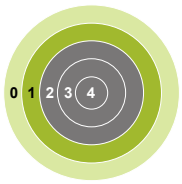
Value	Mode	Description
0	USHFRCO	USHFRCO (clock recovery) is clocking USB
1	HFXO	HFXO clock is used to clock USB
2	HFXOX2	HFXO clock doubler is used to clock USB
3	HFRCO	HFRCO clock is used to clock USB
4	LFXO	LFXO clock is used to clock USB
5	LFRCO	LFRCO clock is used to clock USB

10.5.61 CMU_USBCRCTRL - USB Clock Recovery Control

Offset	Bit Position																													
0x1F4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																															0	0
Access																															RW	RW
Name																															USBLSCRMD	USBCREN

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	USBLSCRMD	0	RW	**Low Speed Clock Recovery Mode**
	This bit must be set to 1 if clock recovery is used when operating as a Low Speed USB device.			
0	USBCREN	0	RW	**Clock Recovery Enable**
	This bit enables the USB clock recovery feature.			

11. SMU - Security Management Unit



Quick Facts

What?

The Security Management Unit (SMU) forms the control and status/reporting component of bus-level security in the EFM32 Giant Gecko 12.

Why?

Enables a robust and low-energy security solution at the system level

How?

Hardware context switching and access control provided via BLS components.

11.1 Introduction

The Security Management Unit (SMU) peripheral adds hardware access control over all of the MCU peripherals that are managed by low level firmware integrated into a Real Timer Operating System (RTOS). The SMU is used in conjuntion with the Cortex-M operating modes (privileged and non-privileged) and the Memory Protection Unit (MPU). The EFM32 Giant Gecko 12 MCUs include the ARM v7-M MPU that defines configurable access parameters to regions within the entire CPU memory map. The MPU is not covered in detail in this reference manual. For a complete description of the MPU registers etc, consult the ARM v7-M Architecture Reference Manual. The MPU can define up to 8 regions of varying sizes within the memory map, with each region also being able to be split into 8 equal sub-regions. Using these regions, firmware can define rules that enforce privileged and non-privileged accesses to different memory locations. For example, sections of flash can be marked as priviliged access, whereas other areas within the flash can be marked as having non-privileged mode acess. Only privileged mode regions can access other privileged mode regions. Accesses attempted by a non-privileged region to a privileged region will cause a fault. The access permissions can be extended across the entire memory map including the peripheral region.

The Cortex-M starts up in privileged mode and the MPU is disabled after reset which means all regions in the memory map are accessble to the running application code. For many applications this is sufficient and the MPU remains disabled. However, when using a RTOS the kernel requires protection from user code and will switch to privileged mode and create tasks in non-privileged or thread mode. In addition, security is also a concern, so MCU peripherals should be protected to avoid security holes. Adding peripheral security to systems requires an increased number of MPU regions to protect areas such as the peripheral registers, including bit set/clear and bit banding regions. The defined regions are also dynamic based on the task requirements and in many cases the number of regions required exceeds the number of regions that can be enabled by the MPU.

The SMU is used to extend the access controls of each peripheral beyond the number of regions available using the MPU. The SMU peripheral registers provide the configuration and status bits for the Peripheral Protection Unit (PPU) to the CPU. The PPU is the underlying hardware component that operates on the low level bus interfaces within the SoC to derive the status for each peripheral .

11.2 Features

The main features of the SMU are as follows:

- Contains control and status registers for hardware bus level component instances (e.g., the PPU)
- Simplifies RTOS context switching
  - Hardware to complement any software context switching enabled by an MPU
  - Hardware-enforced access control extends capability of the v7-M MPU regions
  - One bit control per peripheral reduces software overhead while dynamically modifying access permissions
- A configurable interrupt line that can be triggered from peripheral access fault events

11.3 Functional Description

An overview of the SMU module within the system is shown in Figure 11.1 Bus-Level Security System View on page 471.

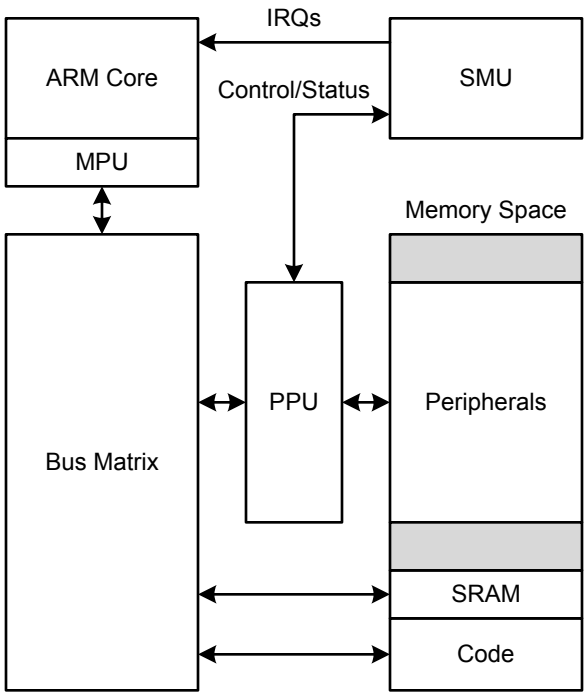


Figure 11.1. Bus-Level Security System View

11.3.1 PPU - Peripheral Protection Unit

The number of peripheral memory regions on the device exceeds the number of configurable regions available using the MPU. While it is possible to manage finer granularity of memory security through software, the PPU provides a hardware solution for fine-grained peripheral-level protection to eliminate the performance degradation associated with a partially software-managed solution.

The PPU provides a hardware access barrier to any peripheral that is configured to be protected. When an attempt is made to access a peripheral without the required privilege level, the PPU detects the fault and intercepts the access. No write or read of the peripheral register space occurs, and an all-zero value is returned if the access is a read. See 11.3.2.2 PPU Control for more details on how access faults are reported to the CPU.

Note: The CPU is the only system bus master in the EFM32 Giant Gecko 12 that can trigger access faults. All other masters are given full access privileges and have no configurable context switching enabled.

11.3.2 Programming Model

The SMU does not provide any access control out of reset and needs to be configured by software. SMU access controls should be configured along with the MPU configuration. This is typically performed in a bootloader or other low level RTOS kernel/supervisor code prior to user code or other non-privileged code execution. At least one MPU region will be allocated to the entire peripheral region as a full access region (0x4000_0000 - 0x4006_FFFF). An RTOS kernel/supervisor can dynamically allocate peripheral accessibility by maintaining the hardware and software contexts available to each task. In the chart below there are mutiple tasks and the system switches between Task A and Task B via the RTOS handler. There are 16 peripherals shown in the example split between two regions. Task A has rights to access peripherals 0, 1, 4, 5 and 7, whereas task B has rights to access the complement of A (2, 3, and 6). After a Task B IRQ, the privileged OS handler is entered which signals the supervisor to reprogram the regions using the SMU based on an access control list. Control is then handed to Task B in non-privileged mode.

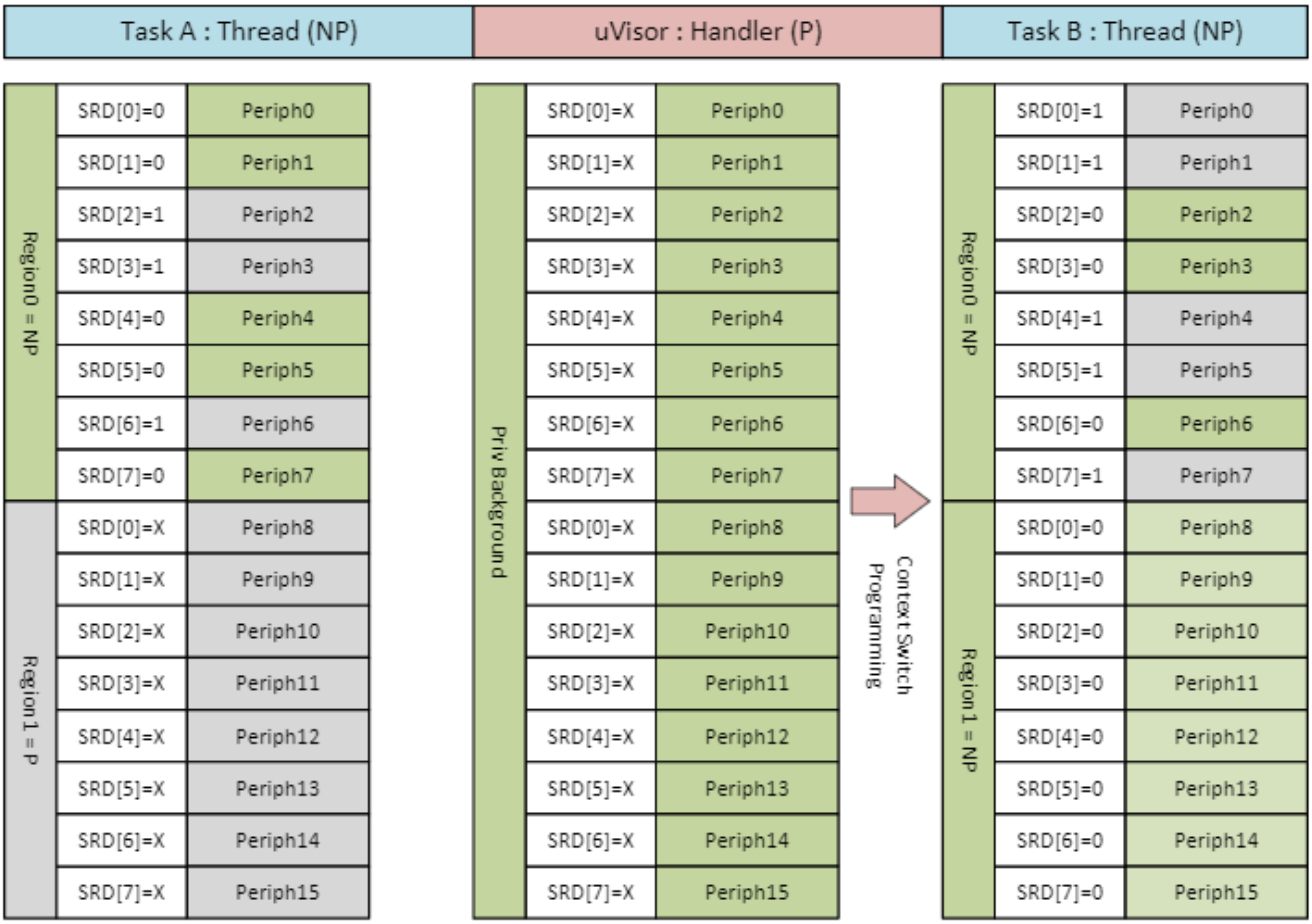


Figure 11.2. Peripheral Access Control Example

All hardware protections happen immediately in response to SMU configuration register writes without any latency cycles. However, since software instructions may be optimized or pipelined, it is important to make sure that software memory barrier instructions are used as needed after any SMU re-configuration before moving on or changing contexts. This ensures that the hardware context switch has taken full effect.

For the remainder of this section, the programming model is split into general SMU controls and component-specific controls (e.g., PPU).

11.3.2.1 Interrupt Control/Status

The SMU follows the standard EFM32 Giant Gecko 12 interrupt programming model with SMU_IF/IFS/IFC/IEN registers.

There is one interrupt bit PPUPRIV that will trigger on privilege faults detected by the PPU. Such fault mechanisms are configured as specified in 11.3.2.2 PPU Control.

11.3.2.2 PPU Control

The PPU_CTRL register provides an ENABLE bit that allows bypassing all PPU checking when set to 0. In this case, the rest of the PPU registers have no effect, and no access faults will occur. This is the reset state of the SMU.

When the ENABLE bit of PPU_CTRL register is asserted, access protection is configured on a peripheral-by-peripheral basis using the SMU_PPUPATDx register(s). Setting a bit in the SMU_PPUPATDx register to one configures the corresponding peripheral controlled by that bit to privileged access only. The single bit mode control for each peripheral provides fast hardware context switching for peripheral sharing, while still supporting fast software context switches for task-based CPU context switching.

Note: The SMU itself is a peripheral which has protection afforded by the PPU. A proper security/privilege context configuration requires setting of the SMU's access control bits properly at startup so that only a top-level task (e.g., a uVisor from ARM) can perform security/privilege context switches.

When a peripheral has access protection configured and the peripheral is accessed with invalid privilege credentials, then an access fault occurs. The corresponding interrupt flag in SMU_IF is asserted and the ID of the peripheral for which an unpriviliged access was attempted is captured in the PERIPHID bit-field of the PPU Fault Status register (SMU_PPUFS). This peripheral ID is held stable until all PPU interrupt flags are cleared to ensure that the first unprivileged access that caused the fault is not overwritten due to subsequent faults before being acknowledged by software.

Note: In the case of simultaneously occurring faults (which may be possible in some systems), only one of the faults' peripheral IDs will be captured. There is no inherent peripheral priority defined that would result in one peripheral being recognized before another.

11.4 Register Map

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

Offset	Name	Type	Description
0x00C	SMU_IF	R	Interrupt Flag Register
0x010	SMU_IFS	W1	Interrupt Flag Set Register
0x014	SMU_IFC	(R)W1	Interrupt Flag Clear Register
0x018	SMU_IEN	RW	Interrupt Enable Register
0x040	SMU_PPUCTRL	RW	PPU Control Register
0x050	SMU_PPUPATD0	RW	PPU Privilege Access Type Descriptor 0
0x054	SMU_PPUPATD1	RW	PPU Privilege Access Type Descriptor 1
0x058	SMU_PPUPATD2	RW	PPU Privilege Access Type Descriptor 2
0x090	SMU_PPUFS	R	PPU Fault Status

11.5 Register Description

11.5.1 SMU_IF - Interrupt Flag 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																																0
Access																																R
Name																																PPUPRIV

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	PPUPRIV	0	R	**PPU Privilege Interrupt Flag**
	Triggered when a privilege fault occurs in the Peripheral Protection Unit			

11.5.2 SMU_IFS - Interrupt Flag 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																																0
Access																																W1
Name																																PPUPRIV

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	PPUPRIV	0	W1	**Set PPUPRIV Interrupt Flag**
	Write 1 to set the PPUPRIV interrupt flag			

11.5.3 SMU_IFC - Interrupt Flag 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																																0
Access																																(R)W1
Name																																PPUPRIV

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	PPUPRIV	0	(R)W1	**Clear PPUPRIV Interrupt Flag**
	Write 1 to clear the PPUPRIV interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

11.5.4 SMU_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																																0
Access																																RW
Name																																PPUPRIV

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	PPUPRIV	0	RW	**PPUPRIV Interrupt Enable**
	Enable/disable the PPUPRIV interrupt			

11.5.5 SMU_PPUCTRL - PPU Control 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																																0
Access																																RW
Name																																ENABLE

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	ENABLE	0	RW	

Set to enable checking of peripheral access

Value	Description
0	Privilege/Security-level checking completely bypassed in the PPU
1	Behavior controlled by PPU_PATD

11.5.6 SMU_PPUPATD0 - PPU Privilege Access Type Descriptor 0

Set peripheral bits to 1 to mark as privileged access only

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				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
Name				LEUART1	LEUART0	LETIMER1	LETIMER0	LESENSE	LDMA	LCD	MSC	IDAC0	I2C1	I2C0	GPIO	GPCRC	FPUEH	EMU	EBI	PRS	VDAC0	CSEN	CRYPTO0	CRYOTIMER	CMU	CAN1	CAN0	ADC1	ADC0	ACMP2	ACMP1	ACMP0

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	LEUART1	0	RW	**Low Energy UART 1 access control bit**
	Access control only for LEUART1			
27	LEUART0	0	RW	**Low Energy UART 0 access control bit**
	Access control only for LEUART0			
26	LETIMER1	0	RW	**Low Energy Timer 1 access control bit**
	Access control only for LETIMER1			
25	LETIMER0	0	RW	**Low Energy Timer 0 access control bit**
	Access control only for LETIMER0			
24	LESENSE	0	RW	**Low Energy Sensor Interface access control bit**
	Access control only for LESENSE			
23	LDMA	0	RW	**Linked Direct Memory Access Controller access control bit**
	Access control only for LDMA			
22	LCD	0	RW	**Liquid Crystal Display Controller access control bit**
	Access control only for LCD			
21	MSC	0	RW	**Memory System Controller access control bit**
	Access control only for MSC			
20	IDAC0	0	RW	**Current Digital to Analog Converter 0 access control bit**
	Access control only for IDAC0			
19	I2C1	0	RW	**I2C 1 access control bit**
	Access control only for I2C1			
18	I2C0	0	RW	**I2C 0 access control bit**
	Access control only for I2C0			
17	GPIO	0	RW	**General purpose Input/Output access control bit**
	Access control only for GPIO			

Bit	Name	Reset	Access	Description
16	GPCRC	0	RW	**General Purpose CRC access control bit**
	Access control only for GPCRC			
15	FPUEH	0	RW	**FPU Exception Handler access control bit**
	Access control only for FPUEH			
14	EMU	0	RW	**Energy Management Unit access control bit**
	Access control only for EMU			
13	EBI	0	RW	**External Bus Interface access control bit**
	Access control only for EBI			
12	PRS	0	RW	**Peripheral Reflex System access control bit**
	Access control only for PRS			
11	VDAC0	0	RW	**Digital to Analog Converter 0 access control bit**
	Access control only for VDAC0			
10	CSEN	0	RW	**Capacitive touch sense module access control bit**
	Access control only for CSEN			
9	CRYPTO0	0	RW	**Advanced Encryption Standard Accelerator access control bit**
	Access control only for CRYPTO0			
8	CRYOTIMER	0	RW	**CRYOTIMER access control bit**
	Access control only for CRYOTIMER			
7	CMU	0	RW	**Clock Management Unit access control bit**
	Access control only for CMU			
6	CAN1	0	RW	**CAN 1 access control bit**
	Access control only for CAN1			
5	CAN0	0	RW	**CAN 0 access control bit**
	Access control only for CAN0			
4	ADC1	0	RW	**Analog to Digital Converter 0 access control bit**
	Access control only for ADC1			
3	ADC0	0	RW	**Analog to Digital Converter 0 access control bit**
	Access control only for ADC0			
2	ACMP2	0	RW	**Analog Comparator 2 access control bit**
	Access control only for ACMP2			
1	ACMP1	0	RW	**Analog Comparator 1 access control bit**
	Access control only for ACMP1			
0	ACMP0	0	RW	**Analog Comparator 0 access control bit**
	Access control only for ACMP0			

11.5.7 SMU_PPUPATD1 - PPU Privilege Access Type Descriptor 1

Set peripheral bits to 1 to mark as privileged access only

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						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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
Name						WTIMER1	WTIMER0	WDOG1	WDOG0	USB	USART4	USART3	USART2	USART1	USART0	UART1	UART0	TRNG0	TIMER3	TIMER2	TIMER1	TIMER0	SMU	SDIO	RTCC	RTC	RMU	QSPI0	PDM	PCNT2	PCNT1	PCNT0

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	WTIMER1	0	RW	**Wide Timer 0 access control bit**
	Access control only for WTIMER1			
25	WTIMER0	0	RW	**Wide Timer 0 access control bit**
	Access control only for WTIMER0			
24	WDOG1	0	RW	**Watchdog access control bit**
	Access control only for WDOG1			
23	WDOG0	0	RW	**Watchdog access control bit**
	Access control only for WDOG0			
22	USB	0	RW	**Universal Serial Bus Interface access control bit**
	Access control only for USB			
21	USART4	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 4 access control bit**
	Access control only for USART4			
20	USART3	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 3 access control bit**
	Access control only for USART3			
19	USART2	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 2 access control bit**
	Access control only for USART2			
18	USART1	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 1 access control bit**
	Access control only for USART1			
17	USART0	0	RW	**Universal Synchronous/Asynchronous Receiver/Transmitter 0 access control bit**
	Access control only for USART0			
16	UART1	0	RW	**Universal Asynchronous Receiver/Transmitter 1 access control bit**
	Access control only for UART1			

Bit	Name	Reset	Access	Description
15	UART0	0	RW	**Universal Asynchronous Receiver/Transmitter 0 access control bit**
	Access control only for UART0			
14	TRNG0	0	RW	**True Random Number Generator 0 access control bit**
	Access control only for TRNG0			
13	TIMER3	0	RW	**Timer 3 access control bit**
	Access control only for TIMER3			
12	TIMER2	0	RW	**Timer 2 access control bit**
	Access control only for TIMER2			
11	TIMER1	0	RW	**Timer 1 access control bit**
	Access control only for TIMER1			
10	TIMER0	0	RW	**Timer 0 access control bit**
	Access control only for TIMER0			
9	SMU	0	RW	**Security Management Unit access control bit**
	Access control only for SMU			
8	SDIO	0	RW	**SDIO Controller access control bit**
	Access control only for SDIO			
7	RTCC	0	RW	**Real-Time Counter and Calendar access control bit**
	Access control only for RTCC			
6	RTC	0	RW	**Real-Time Counter access control bit**
	Access control only for RTC			
5	RMU	0	RW	**Reset Management Unit access control bit**
	Access control only for RMU			
4	QSPI0	0	RW	**Quad-SPI access control bit**
	Access control only for QSPI0			
3	PDM	0	RW	**PDM Interface access control bit**
	Access control only for PDM			
2	PCNT2	0	RW	**Pulse Counter 2 access control bit**
	Access control only for PCNT2			
1	PCNT1	0	RW	**Pulse Counter 1 access control bit**
	Access control only for PCNT1			
0	PCNT0	0	RW	**Pulse Counter 0 access control bit**
	Access control only for PCNT0			

11.5.8 SMU_PPUPATD2 - PPU Privilege Access Type Descriptor 2

Set peripheral bits to 1 to mark as privileged access only

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																																
Access																																
Name																																

Bit	Name	Reset	Access	Description
31:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

11.5.9 SMU_PPUFS - PPU Fault Status

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																													0x00		
Access																													R		
Name																													PERIPHID		

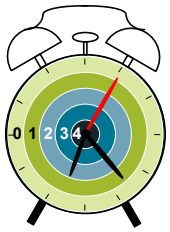
Bit	Name	Reset	Access	Description
31:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	PERIPHID	0x00	R	

Holds the peripheral ID of the first peripheral that was accessed resulting in an access fault. This ID is not valid unless one of the PPU interrupt flags is set. Any other access faults that occur are not captured until all the PPU interrupt flags are cleared

Value	Mode	Description
0	ACMP0	Analog Comparator 0
1	ACMP1	Analog Comparator 1
2	ACMP2	Analog Comparator 2
3	ADC0	Analog to Digital Converter 0
4	ADC1	Analog to Digital Converter 0
5	CAN0	CAN 0
6	CAN1	CAN 1
7	CMU	Clock Management Unit
8	CRYOTIMER	CRYOTIMER
9	CRYPTO0	Advanced Encryption Standard Accelerator
10	CSEN	Capacitive touch sense module
11	VDAC0	Digital to Analog Converter 0
12	PRS	Peripheral Reflex System
13	EBI	External Bus Interface
14	EMU	Energy Management Unit
15	FPUEH	FPU Exception Handler
16	GPCRC	General Purpose CRC
17	GPIO	General purpose Input/Output
18	I2C0	I2C 0
19	I2C1	I2C 1
20	IDAC0	Current Digital to Analog Converter 0
21	MSC	Memory System Controller

Bit	Name	Reset	Access	Description
	22	LCD		Liquid Crystal Display Controller
	23	LDMA		Linked Direct Memory Access Controller
	24	LESENSE		Low Energy Sensor Interface
	25	LETIMER0		Low Energy Timer 0
	26	LETIMER1		Low Energy Timer 1
	27	LEUART0		Low Energy UART 0
	28	LEUART1		Low Energy UART 1
	32	PCNT0		Pulse Counter 0
	33	PCNT1		Pulse Counter 1
	34	PCNT2		Pulse Counter 2
	35	PDM		PDM Interface
	36	QSPI0		Quad-SPI
	37	RMU		Reset Management Unit
	38	RTC		Real-Time Counter
	39	RTCC		Real-Time Counter and Calendar
	40	SDIO		SDIO Controller
	41	SMU		Security Management Unit
	42	TIMER0		Timer 0
	43	TIMER1		Timer 1
	44	TIMER2		Timer 2
	45	TIMER3		Timer 3
	46	TRNG0		True Random Number Generator 0
	47	UART0		Universal Asynchronous Receiver/Transmitter 0
	48	UART1		Universal Asynchronous Receiver/Transmitter 1
	49	USART0		Universal Synchronous/Asynchronous Receiver/Transmitter 0
	50	USART1		Universal Synchronous/Asynchronous Receiver/Transmitter 1
	51	USART2		Universal Synchronous/Asynchronous Receiver/Transmitter 2
	52	USART3		Universal Synchronous/Asynchronous Receiver/Transmitter 3
	53	USART4		Universal Synchronous/Asynchronous Receiver/Transmitter 4
	54	USB		Universal Serial Bus Interface
	55	WDOG0		Watchdog
	56	WDOG1		Watchdog
	57	WTIMER0		Wide Timer 0
	58	WTIMER1		Wide Timer 0

12. RTCC - Real Time Counter and Calendar



Quick Facts

What?

The Real Time Counter and Calendar (RTCC) is a 32-bit counter ensuring timekeeping in low energy modes. The RTCC also includes a calendar mode for easy time and date keeping. In addition, the RTCC includes 128 bytes of general purpose retention data, allowing persistent data storage in all energy modes except EM4 Shutoff.

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.

12.1 Introduction

The Real Time Counter and Calendar (RTCC) contains a 32-bit counter/calendar in combination with a 15-bit pre-counter to allow flexible prescaling of the main counter. The RTCC is available in all energy modes except EM4 Shutoff.

Three individually configurable Capture/Compare channels are available in the RTCC. These can be used to trigger interrupts, generate PRS signals, capture system events, and to wake the device up from a low energy mode. The RTCC also includes 128 bytes of general purpose storage and a Binary Coded Decimal (BCD) calendar mode, enabling easy time and date keeping.

12.2 Features

- 32-bit Real Time Counter.
- 15-bit pre-counter, for flexible frequency scaling or for use as an independent counter.
- EM4 Hibernate operation and wakeup.
- 128 byte general purpose retention data.
- Oscillator failure detection.
- Backup mode Timestamp.
- Can continue through system reset; only reset by power loss, pin, or software reset.
- Calendar mode.
  - BCD encoding.
  - Three programmable alarms.
  - Leap year correction.
- Three Capture/Compare registers.
  - Capture of PRS events from other parts of the system.
  - Compare match or input capture can trigger interrupts.
  - Compare register 1, RTCC_CC1_CCV can be used as a top value for the main counter.
  - Compare register 0, RTCC_CC0_CCV 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).

12.3 Functional Description

The RTCC is a 32-bit up-counter with three Capture/Compare channels. In addition, the RTCC includes a 15-bit pre-counter which can be used as an independent counter or to prescale the main counter. An overview of the RTCC module is shown in Figure 12.1 RTCC Overview on page 485.

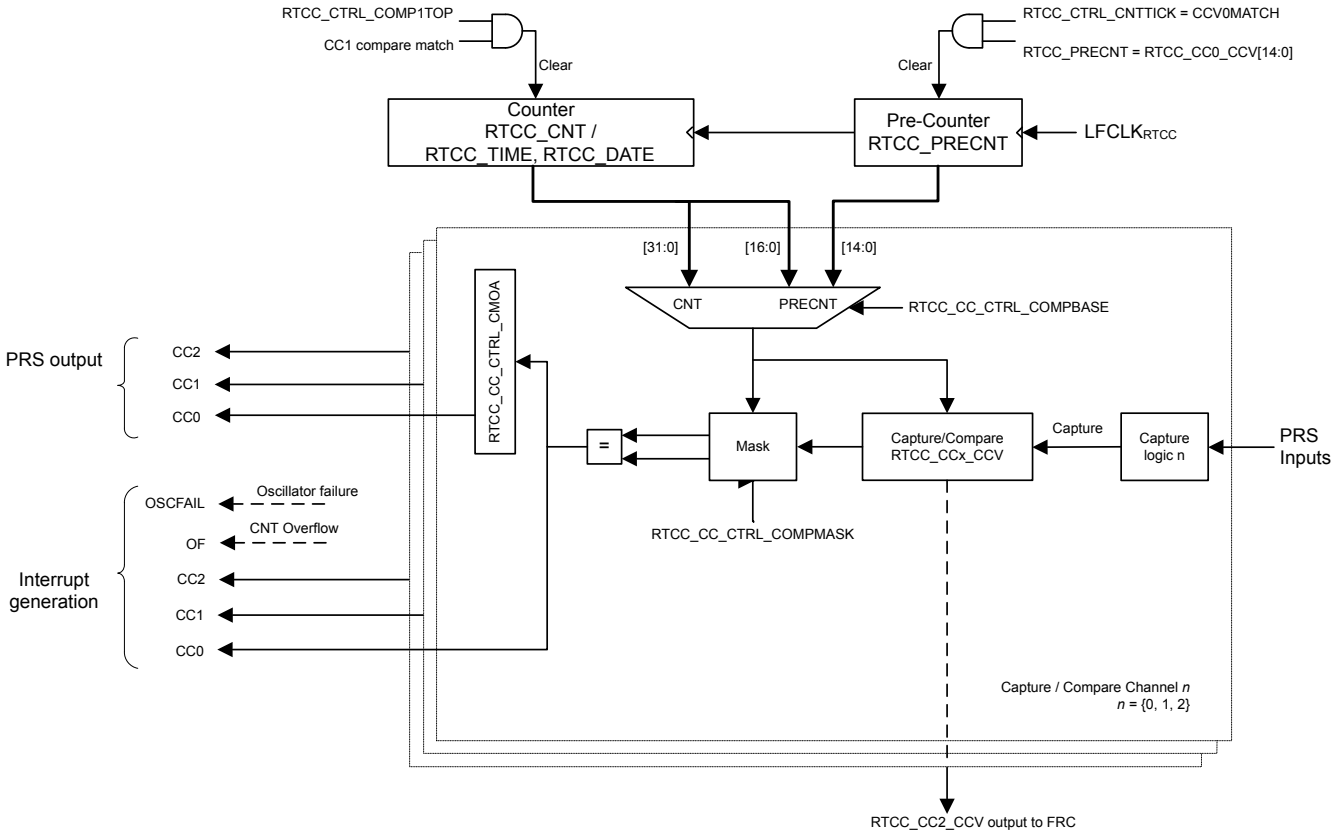


Figure 12.1. RTCC Overview

12.3.1 Counter

The RTCC consists of two counters; the 32-bit main counter, RTCC_CNT (RTCC_TIME and RTCC_DATE in calendar mode), 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 12.3.2 Capture/Compare Channels for details on how to configure the Capture/Compare channels for use with the pre-counter.

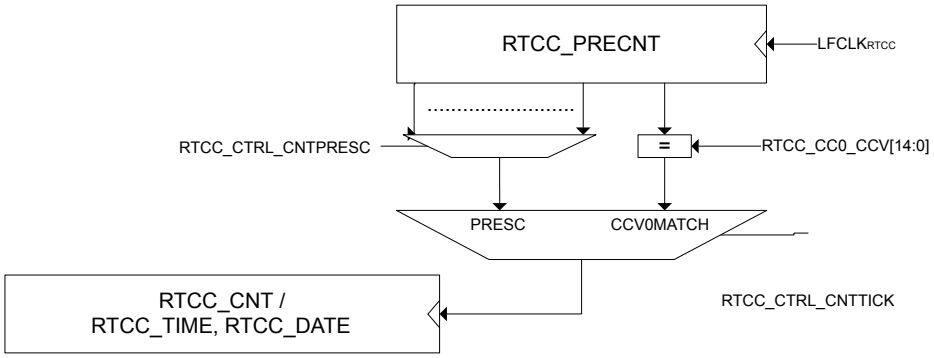


Figure 12.2. RTCC counters

The RTCC is enabled by setting the ENABLE bit in RTCC_CTRL. When the RTCC is enabled, the pre-counter (RTCC_PRECNT) increments upon each positive clock edge of $LFCLK_{RTCC}$. If CNTTICK in RTCC_CTRL is set to PRESC, the pre-counter will continue to count up, wrapping around to zero when it overflows. If CNTTICK in RTCC_CTRL is set to CCV0MATCH, the pre-counter will wrap around when it hits the value configured in RTCC_CC0_CCV.

The main counter of the RTCC, RTCC_CNT, has two modes; normal mode and calendar mode. In normal mode, the main counter is available in RTCC_CNT and increments upon each tick given from the pre-counter. Refer to 12.3.1.1 Normal Mode for a description on how to configure the frequency of these ticks. In calendar mode, the counter value is available in RTCC_TIME and RTCC_DATE, keeping track of seconds, minutes, hours, day of month, day of week, months, and years, all encoded in BCD format. Refer to 12.3.1.2 Calendar Mode for details on this mode. The mode of the main counter is configured in CNTMODE in RTCC_CTRL. The differences between the two modes are summarized below.

- **Normal mode**
  - Incremental counter, RTCC_CNT.
  - RTCC_CCx_CCV used for Capture/Compare value.
- **Calendar mode**
  - BCD counters, RTCC_DATE, RTCC_TIME.
  - RTCC_CCx_TIME and RTCC_CCx_DATE used for Capture/Compare value.

Note: The mode of the RTCC must be configured for CALENDAR mode in RTCC_CTRL_CNTMODE before writing to the mode dependent registers, RTCC_TIME, RTCC_DATE, RTCC_CCx_TIME, and RTCC_CCx_DATE. Writes to these registers when in NORMAL mode will be ignored.

12.3.1.1 Normal Mode

The main counter can receive a tick based on different tappings from the pre-counter, allowing the ticks to be power of 2 divisions of the $LFCLK_{RTCC}$. For more accurate configuration of the tick frequency, RTCC_CC0_CCV[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 12.1 RTCC Resolution Vs Overflow, F_{LFCLK} = 32768 Hz on page 487 summarizes the resolutions available when using a 32768 Hz oscillator as source for $LFCLK_{RTCC}$.

Table 12.1. 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_CCV + 1)/F_{LFCLK} s	2^{32}*T_{CNT} seconds
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_CTRL, 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_CCV). Before 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 (RTCC_CTRL_MODE != DISABLED) may cause unintended operation (e.g. if RTCC_CNT > RTCC_CC1_CCV, RTCC_CNT will wrap when reaching 0xFFFFFFFF rather than RTCC_CC1_CCV).

Note: If the RTCC is being reconfigured, and capture compare channel 1 has previously been used, a CCV1TOP wrap event might be pending. This would lead to the first tick of the main counter being a wrap to 0. To clear any pending wrap events, use the following procedure before reconfiguring the RTCC:

1. RTCC->CC[1].CTRL = RTCC_CC_CTRL_MODE_OFF;
2. RTCC->CTRL = RTCC_CTRL_CNTTICK_PRESC | RTCC_CTRL_CNTMODE_NORMAL | RTCC_CTRL_ENABLE;
3. rtcc_cnt_pre = RTCC->CNT;
4. while(RTCC->CNT == rtcc_cnt_pre);
5. Reconfigure the RTCC

12.3.1.2 Calendar Mode

The RTCC includes a calendar mode which implements time and date decoding in hardware. Calendar mode is enabled by configuring CNTMODE in RTCC_CTRL to CALENDAR. When in calendar mode, the counter value is available in RTCC_TIME and RTCC_DATE. RTCC_TIME shows seconds, minutes, and hours while RTCC_DATE shows day of month, month, year, and day of week. RTCC_TIME and RTCC_DATE are encoded in BCD format. In calendar mode, the pre-counter should be configured to give ticks with a period of one second, i.e. RTCC_CTRL_CNTTICK should be set to PRESC, and the CNTPRESC bitfield of the RTCC_CTRL register should be set to DIV32768 if a 32768 Hz clock source is used.

In calendar mode, the time and date registers of the capture compare channels, RTCC_CCx_TIME and RTCC_CCx_DATE, are used to set compare values. Compare values can be set on seconds, minutes, hours, days, and months. Whether day of week or day of month is used for a Capture/Compare channel, it is configured in RTCC_CCx_CTRL_DAYCC of the respective Capture/Compare channel.

The RTCC will automatically compensate for 28-, 29- (leap year), 30-, and 31-day months. The day of week counter, RTCC_DATE_DAYOW, is a three bit counter incrementing when RTCC_TIME_HOURT overflows, wrapping around every seventh day. Automatic leap year correction, extending the month of February from 28 to 29 days every fourth year is by default enabled, but can be disabled by setting the LYEARCORRDIS bit in RTCC_CTRL. The pseudo-code for leap year correction is as follows:

```
if RTCC_DATE_YEART modulo 2 = 0:
   if RTCC_DATE_YEARU modulo 4 = 0:
      leap_year = true
   else:
      leap_year = false
else:
   if (RTCC_DATE_YEARU + 2) modulo 4 = 0:
      leap_year = true
   else:
      leap_year = false
```

The seconds, minute, hour segments are represented in 24-hour BCD format. The month segments are enumerated as shown in Table 12.2 RTCC calendar enumeration on page 488.

Table 12.2. RTCC calendar enumeration

Month	RTCC_DATE_MONTHT	RTCC_DATE_MONTHU
January	0b0	0b0001
February	0b0	0b0010
March	0b0	0b0011
April	0b0	0b0100
May	0b0	0b0101
June	0b0	0b0110
July	0b0	0b0111
August	0b0	0b1000
September	0b0	0b1001
October	0b1	0b0000
November	0b1	0b0001
December	0b1	0b0010

12.3.1.3 RTCC Initialization

The counters of the RTCC, RTCC_CNT (RTCC_TIME and RTCC_DATE in calendar mode) 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 immediate synchronization scheme, described in 4.3.1 Writing.

Note: Writing to the RTCC_PRECNT register may alter the frequency of the ticks for the RTCC_CNT register.

12.3.2 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.

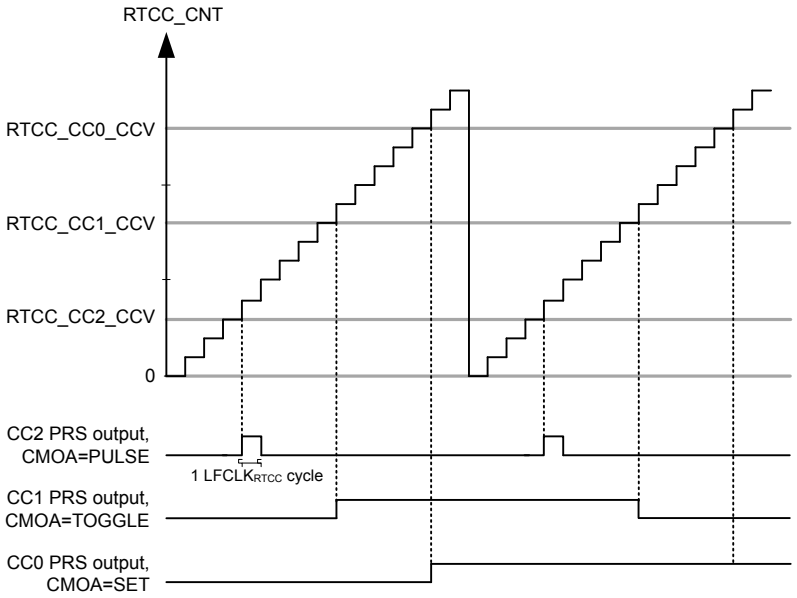


Figure 12.3. RTCC Compare match and PRS output illustration

In input capture mode the RTCC_CNT (RTCC_TIME and RTCC_DATE in calendar mode) register is captured into the RTCC_CCx_CCV (RTCC_CCx_TIME and RTCC_CCx_DATE in calendar mode) 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_CCV (RTCC_CCx_TIME and RTCC_CCx_DATE in calendar mode). These values will be compared to the main counter, RTCC_CNT (RTCC_TIME and RTCC_DATE in calendar mode), or a mixture of the main counter and the pre-counter, as illustrated in Figure 12.4 RTCC Compare base illustration on page 491. Compare base for the capture compare channels is set by configuring COMPBASE in RTCC_CCx_CTRL.

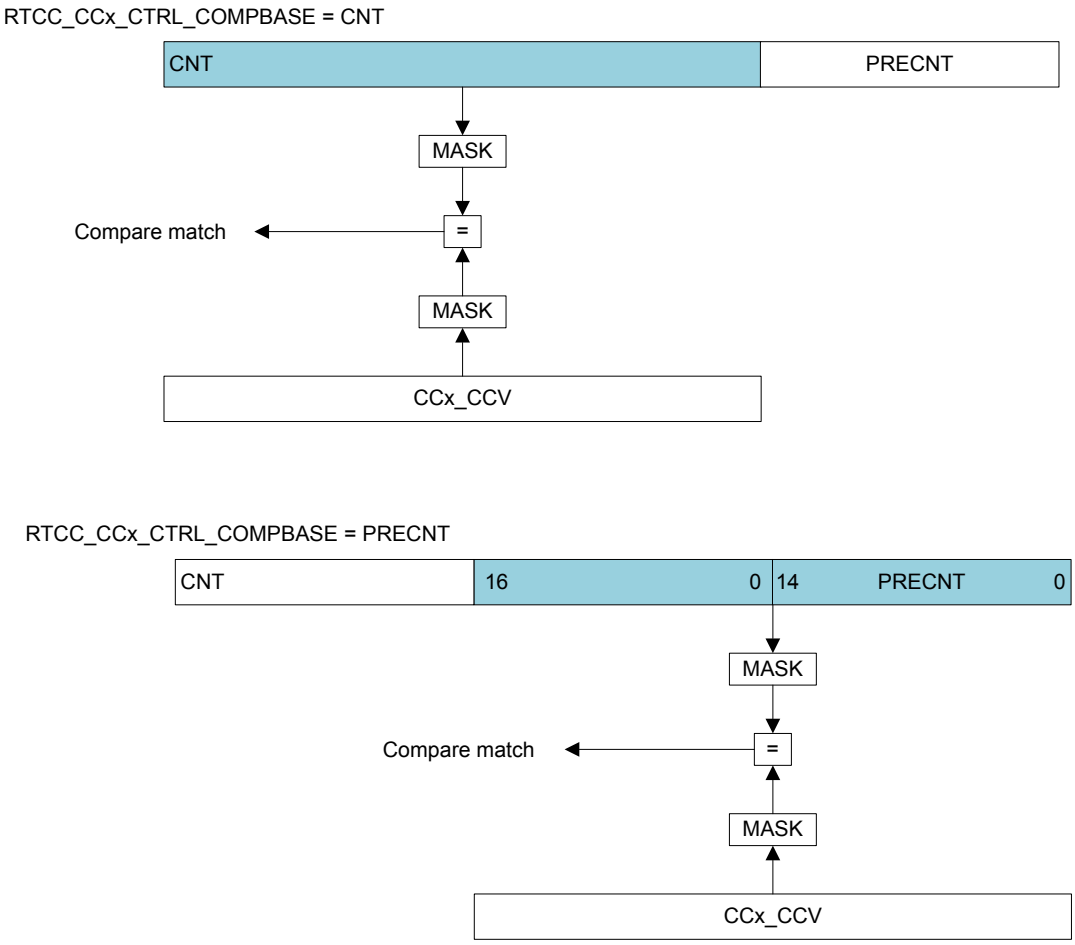


Figure 12.4. RTCC Compare base illustration

Table 12.3 RTCC Capture/Compare Subjects on page 491 summarizes which registers being subject to comparison for different configurations of RTCC_CTRL_CNTMODE and RTCC_CCx_CTRL_COMPBASE.

Table 12.3. RTCC Capture/Compare Subjects

RTCC_CTRL_CNTMODE	NORMAL	CALENDAR
RTCC_CCx_CTRL_COMPBASE = CNT	RTCC_CNT vs. RTCC_CCx_CCV	RTCC_TIME vs. RTCC_CCx_TIME and RTCC_DATE vs. RTCC_CCx_DATE
RTCC_CCx_CTRL_COMPBASE = PRECNT	{RTCC_CNT[16:0],RTCC_PRECNT[14:0]} vs. RTCC_CCx_CCV	RTCC_PRECNT vs. RTCC_CCx_CCV[14:0]

Figure 12.5 RTCC Compare in calendar mode, COMPBASE = CNT on page 492 illustrates how the compare events are evaluated when in calendar mode with RTCC_CCx_CTRL_COMPBASE = CNT. The SECU, SECT, MINU, MINT, HOURU, HOURT, MONTHU, and MONTHT bitfields in RTCC_CCx_TIME and RTCC_CCx_DATE are compared to the corresponding bitfields in RTCC_DATE and RTCC_TIME. The DAYU and DAYT bitfields in RTCC_CCx_DATE will be compared to {RTCC_DATE_DAYOMT, RTCC_DATE_DAYOMU} if DAYCC in RTCC_CCx_CTRL is set to MONTH. If DAYCC in RTCC_CCx_CTRL is set to WEEK, the DAYU and DAYT bitfields in RTCC_CCx_DATE will be compared to {0b000, RTCC_DATE_DAYOW}.

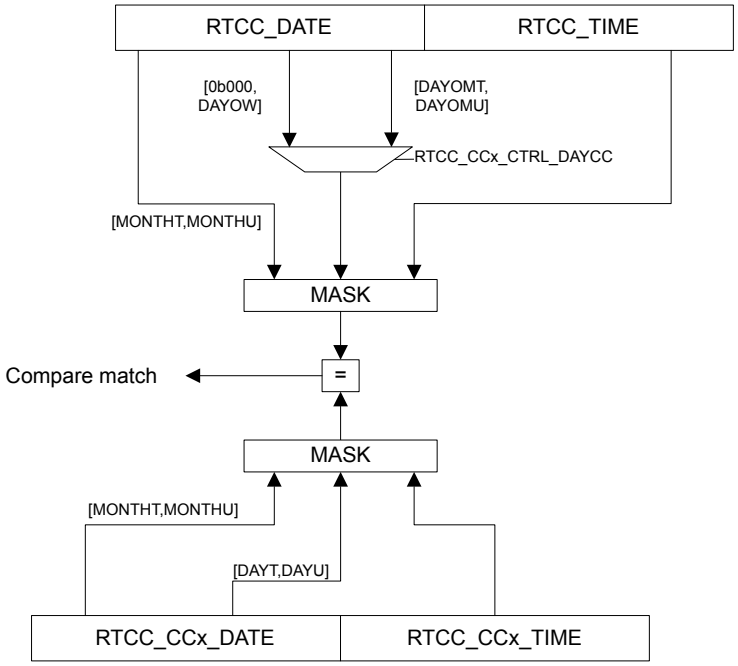


Figure 12.5. RTCC Compare in calendar mode, COMPBASE = CNT

To generate periodically recurring events, it is possible to mask out parts of the compare match values. By configuring COMPMASK in RTCC_CCx_CTRL, parts of the compare values will be masked out, limiting which part of the compare register being subject to comparison with the counter. Figure 12.6 RTCC Compare mask illustration, COMPMASK=11 on page 492 illustrates the effect of COMPMASK when in normal mode and calendar mode.

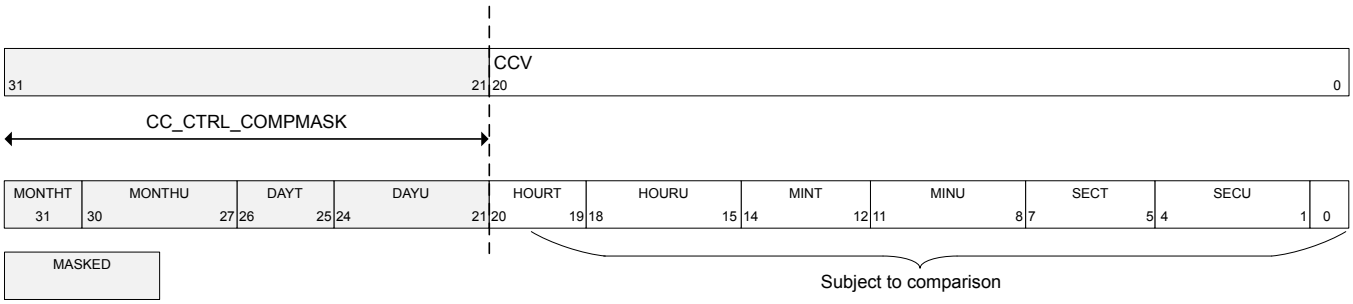


Figure 12.6. RTCC Compare mask illustration, COMPMASK=11

Upon a compare match, the respective Capture/Compare interrupt flag CCx is set. Additionally, the event selected by the CMOA setting is generated on the corresponding PRS output. This is illustrated in Figure 12.3 RTCC Compare match and PRS output illustration on page 490.

12.3.3 Interrupts and PRS Output

The RTCC has one interrupt for each of its 3 Capture/Compare channels, CC0, CC1, and CC2. Each Capture/Compare channel has a PRS output with configurable actions upon compare match.

The interrupt flag CNTTICK is set each time the main counter receives a tick (each second in calendar mode). In calendar mode, there are also interrupt flags being set each minute, hour, day, week, and month.

Upon oscillator failure detection, the OSCFAIL flag will be set.

12.3.3.1 Main Counter Tick PRS Output

To output the ticks for the main counter on PRS, it is possible to use a Capture/Compare channel and mask all the bits, i.e. RTCC_CCx_CTRL_COMPBASE=CNT and RTCC_CCx_CTRL_COMPMASK=31. PRS output of main counter ticks does not work if the main counter is not prescaled.

Note: To be able to mask all bits in the main counter, RTCC_CTRL_CNTMODE has to be set to CALENDAR. In NORMAL mode, the least significant bit can not be masked out.

12.3.4 Energy Mode Availability

The RTCC is available in all energy modes except EM4 Shutoff. To enable RTCC operation in EM4 Hibernate, the EMU_EM4CTRL register in the EMU has to be configured. Any enabled RTCC interrupt will wake the system up from EM4 Hibernate; if EM4WU in RTCC_EM4WUEN is set. Refer to 9. EMU - Energy Management Unit for details on how to configure the EMU.

12.3.5 Register Lock

To prevent accidental writes to the RTCC registers, the RTCC_LOCKKEY register can be written to any value other than the unlock value. To unlock the register, write the unlock value to RTCC_LOCKKEY. Registers affected by this lock are:

- RTCC_CTRL
- RTCC_PRECNT
- RTCC_CNT
- RTCC_TIME
- RTCC_DATE
- RTCC_IEN
- RTCC_POWERDOWN
- RTCC_CCx_CTRL
- RTCC_CCx_CCV
- RTCC_CCx_TIME
- RTCC_CCx_DATE

12.3.6 Oscillator Failure Detection

To be able to detect OSC failure, the RTCC includes a security mechanism ensuring that at least three OSC cycles are detected within one period of the ULFRCO. If no OSC cycles are detected, the OSCFAIL interrupt flag is set. OSC failure detection is enabled by setting the OSCFDETEN bit in RTCC_CTRL.

12.3.7 Retention Registers

The RTCC includes 32 x 32 bit registers which can be retained in all energy modes except EM4 Shutoff. The registers are accessible through the RETx_REG registers. Retention is by default enabled in EM0 Active through EM4 Hibernate/Shutoff. The registers can be shut off to save power by setting the RAM bit in RTCC_POWERDOWN.

Note: The retention registers are mapped to a RAM instance and have undefined state out of reset.

12.3.8 Timestamp

The RTCC includes functionality for storing a timestamp when the system enters backup mode. The timestamp is stored in the RTCC_CC2_CCV (RTCC_CC2_TIME and RTCC_CC2_DATE in calendar mode) register. Timestamping is enabled by first writing a 1 to CLRSTATUS in RTCC_CMD and then setting BUMODETSEN in RTCC_CTRL. When a timestamp is stored, the BUMODETS bit in RTCC_STATUS is set. To prevent uncontrolled time stamping when entering and exiting backup mode, this status bit has to be cleared before a new timestamp can be stored. Writing a 1 to CLRSTATUS in RTCC_CMD clears BUMODETS.

12.3.9 Debug Session

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.

12.4 Register Map

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

Offset	Name	Type	Description
0x000	RTCC_CTRL	RW	Control Register
0x004	RTCC_PRECNT	RWH	Pre-Counter Value Register
0x008	RTCC_CNT	RWH	Counter Value Register
0x00C	RTCC_COMBCNT	R	Combined Pre-Counter and Counter Value Register
0x010	RTCC_TIME	RWH	Time of Day Register
0x014	RTCC_DATE	RWH	Date Register
0x018	RTCC_IF	R	RTCC Interrupt Flags
0x01C	RTCC_IFS	W1	Interrupt Flag Set Register
0x020	RTCC_IFC	(R)W1	Interrupt Flag Clear Register
0x024	RTCC_IEN	RW	Interrupt Enable Register
0x028	RTCC_STATUS	R	Status Register
0x02C	RTCC_CMD	W1	Command Register
0x030	RTCC_SYNCBUSY	R	Synchronization Busy Register
0x034	RTCC_POWERDOWN	RW	Retention RAM Power-down Register
0x038	RTCC_LOCK	RWH	Configuration Lock Register
0x03C	RTCC_EM4WUEN	RW	Wake Up Enable
0x040	RTCC_CC0_CTRL	RW	CC Channel Control Register
0x044	RTCC_CC0_CCV	RWH	Capture/Compare Value Register
0x048	RTCC_CC0_TIME	RWH	Capture/Compare Time Register
0x04C	RTCC_CC0_DATE	RWH	Capture/Compare Date Register
0x050	RTCC_CC1_CTRL	RW	CC Channel Control Register
0x054	RTCC_CC1_CCV	RWH	Capture/Compare Value Register
0x058	RTCC_CC1_TIME	RWH	Capture/Compare Time Register
0x05C	RTCC_CC1_DATE	RWH	Capture/Compare Date Register
0x060	RTCC_CC2_CTRL	RW	CC Channel Control Register
0x064	RTCC_CC2_CCV	RWH	Capture/Compare Value Register
0x068	RTCC_CC2_TIME	RWH	Capture/Compare Time Register
0x06C	RTCC_CC2_DATE	RWH	Capture/Compare Date Register
0x104	RTCC_RET0_REG	RW	Retention Register
...	RTCC_RETx_REG	RW	Retention Register
0x180	RTCC_RET31_REG	RW	Retention Register

12.5 Register Description

12.5.1 RTCC_CTRL - Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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															0	0	0	0		0	0x0						0	0		0		0
Access															RW	RW	RW	RW		RW	RW						RW	RW		RW		RW
Name															LYEARCORRDIS	CNTMODE	OSCFDETEN	BUMODETSEN		CNTTICK	CNTPRESC						CCV1TOP	PRECCV0TOP		DEBUGRUN		ENABLE

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	LYEARCORRDIS	0	RW	**Leap Year Correction Disabled**
	When cleared, February has 29 days in leap years. When set, February always has 28 days.			
16	CNTMODE	0	RW	**Main Counter Mode**
	Configure count mode for the main counter.			

Value	Mode	Description
0	NORMAL	The main counter is incremented with 1 for each tick.
1	CALENDAR	The main counter is in calendar mode.

Bit	Name	Reset	Access	Description
15	OSCFDETEN	0	RW	**Oscillator Failure Detection Enable**
	When set, the OSCFAIL interrupt flag will be set if no ticks are detected on $LFCLK_{RTCC}$ within one ULFRCO cycle.			
14	BUMODETSEN	0	RW	**Backup Mode Timestamp Enable**
	When set, the RTCC will store its counter value in the RTCC_CC2_CCV register upon backup mode entry.			
13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	CNTTICK	0	RW	**Counter Prescaler Mode**
	Select whether the main counter should tick on RTCC_CC0_CCV[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.
1	CCV0MATCH	CNT register ticks when PRECNT matches RTCC_CC0_CCV[14:0]

Bit	Name	Reset	Access	Description
11:8	CNTPRESC	0x0	RW	**Counter Prescaler Value**
	Configure counting frequency of the CNT register.			

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	DIV1		CLK_{CNT} = $LFECLK_{RTCC}$/1
	1	DIV2		CLK_{CNT} = $LFECLK_{RTCC}$/2
	2	DIV4		CLK_{CNT} = $LFECLK_{RTCC}$/4
	3	DIV8		CLK_{CNT} = $LFECLK_{RTCC}$/8
	4	DIV16		CLK_{CNT} = $LFECLK_{RTCC}$/16
	5	DIV32		CLK_{CNT} = $LFECLK_{RTCC}$/32
	6	DIV64		CLK_{CNT} = $LFECLK_{RTCC}$/64
	7	DIV128		CLK_{CNT} = $LFECLK_{RTCC}$/128
	8	DIV256		CLK_{CNT} = $LFECLK_{RTCC}$/256
	9	DIV512		CLK_{CNT} = $LFECLK_{RTCC}$/512
	10	DIV1024		CLK_{CNT} = $LFECLK_{RTCC}$/1024
	11	DIV2048		CLK_{CNT} = $LFECLK_{RTCC}$/2048
	12	DIV4096		CLK_{CNT} = $LFECLK_{RTCC}$/4096
	13	DIV8192		CLK_{CNT} = $LFECLK_{RTCC}$/8192
	14	DIV16384		CLK_{CNT} = $LFECLK_{RTCC}$/16384
	15	DIV32768		CLK_{CNT} = $LFECLK_{RTCC}$/32768
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CCV1TOP	0	RW	**CCV1 Top Value Enable**
	When set, the counter wraps around on a CC1 event.			
4	PRECCV0TOP	0	RW	**Pre-counter CCV0 Top Value Enable**
	When set, the pre-counter wraps around when PRECNT equals RTCC_CC0_CCV[14:0].			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DEBUGRUN	0	RW	**Debug Mode Run Enable**
	Set this bit to keep the RTCC running during a debug halt.			
	Value			Description
	0			RTCC is frozen in debug mode
	1			RTCC is running in debug mode
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	ENABLE	0	RW	**RTCC Enable**
	Enable the RTCC.			

12.5.2 RTCC_PRECNT - Pre-Counter Value Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																		0x0000														
Access																		RWH														
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	0x0000	RWH	**Pre-Counter Value**
	Gives access to the Pre-counter value of the RTCC.			

12.5.3 RTCC_CNT - Counter Value Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																0x00000000																
Access																RWH																
Name																CNT																

Bit	Name	Reset	Access	Description
31:0	CNT	0x00000000	RWH	**Counter Value**
	Gives access to the main counter value of the RTCC. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = CALENDAR.			

12.5.4 RTCC_COMBCNT - Combined Pre-Counter and Counter Value 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	0x00000																	0x0000													
Access	R																	R													
Name	CNTLSB																	PRECNT													

Bit	Name	Reset	Access	Description
31:15	CNTLSB	0x00000	R	**Counter Value**
	Gives access to the 17 LSBs of the main counter, CNT. Register will be read as zero when RTCC_CTRL_CNTMODE = CALENDAR.			
14:0	PRECNT	0x0000	R	**Pre-Counter Value**
	Gives access to the pre-counter, PRECNT. Register will be read as zero when RTCC_CTRL_CNTMODE = CALENDAR.			

12.5.5 RTCC_TIME - Time of Day Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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			
Access											RWH		RWH					RWH			RWH					RWH			RWH			
Name											HOURT		HOURU					MINT			MINU					SECT			SECU			

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	HOURT	0x0	RWH	**Hours, Tens**
	Shows the tens part of the hour counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
19:16	HOURU	0x0	RWH	**Hours, Units**
	Shows the unit part of the hour counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:12	MINT	0x0	RWH	**Minutes, Tens**
	Shows the tens part of the minute counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
11:8	MINU	0x0	RWH	**Minutes, Units**
	Shows the unit part of the minute counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	SECT	0x0	RWH	**Seconds, Tens**
	Shows the tens part of the second counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
3:0	SECU	0x0	RWH	**Seconds, Units**
	Shows the unit part of the second counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			

12.5.6 RTCC_DATE - Date Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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							0	0x0						0x0		0x0			
Access						RWH			RWH				RWH							RWH	RWH						RWH		RWH			
Name						DAYOW			YEART				YEARU							MONTHT	MONTHU						DAYOMT		DAYOMU			

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	DAYOW	0x0	RWH	**Day of Week**
	Shows the day of week counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
23:20	YEART	0x0	RWH	**Year, Tens**
	Shows the tens part of the year counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
19:16	YEARU	0x0	RWH	**Year, Units**
	Shows the unit part of the year counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	MONTHT	0	RWH	**Month, Tens**
	Shows the tens part of the month counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
11:8	MONTHU	0x0	RWH	**Month, Units**
	Shows the unit part of the month counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	DAYOMT	0x0	RWH	**Day of Month, Tens**
	Shows the tens part of the day of month counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
3:0	DAYOMU	0x0	RWH	**Day of Month, Units**
	Shows the unit part of the day of month counter. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			

12.5.7 RTCC_IF - RTCC Interrupt Flags

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																						0	0	0	0	0	0	0	0	0	0	0
Access																						R	R	R	R	R	R	R	R	R	R	R
Name																						MONTHTICK	DAYOWOF	DAYTICK	HOURTICK	MINTICK	CNTTICK	OSCFAIL	CC2	CC1	CC0	OF

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	MONTHTICK	0	R	**Month Tick**
	Set each time the month counter increments.			
9	DAYOWOF	0	R	**Day of Week Overflow**
	Set each time the day of week counter overflows.			
8	DAYTICK	0	R	**Day Tick**
	Set each time the day counter increments.			
7	HOURTICK	0	R	**Hour Tick**
	Set each time the hour counter increments.			
6	MINTICK	0	R	**Minute Tick**
	Set each time the minute counter increments.			
5	CNTTICK	0	R	**Main Counter Tick**
	Set each time the main counter is updated.			
4	OSCFAIL	0	R	**Oscillator Failure Interrupt Flag**
	Set when an oscillator failure has been detected.			
3	CC2	0	R	**Channel 2 Interrupt Flag**
	Set when a channel 2 event has occurred.			
2	CC1	0	R	**Channel 1 Interrupt Flag**
	Set when a channel 1 event has occurred.			
1	CC0	0	R	**Channel 0 Interrupt Flag**
	Set when a channel 0 event has occurred.			
0	OF	0	R	**Overflow Interrupt Flag**
	Set when a RTCC overflow has occurred.			

12.5.8 RTCC_IFS - Interrupt Flag Set 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																						0	0	0	0	0	0	0	0	0	0	0
Access																						W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																						MONTHTICK	DAYOWOF	DAYTICK	HOURTICK	MINTICK	CNTTICK	OSCFAIL	CC2	CC1	CC0	OF

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	MONTHTICK	0	W1	**Set MONTHTICK Interrupt Flag**
	Write 1 to set the MONTHTICK interrupt flag			
9	DAYOWOF	0	W1	**Set DAYOWOF Interrupt Flag**
	Write 1 to set the DAYOWOF interrupt flag			
8	DAYTICK	0	W1	**Set DAYTICK Interrupt Flag**
	Write 1 to set the DAYTICK interrupt flag			
7	HOURTICK	0	W1	**Set HOURTICK Interrupt Flag**
	Write 1 to set the HOURTICK interrupt flag			
6	MINTICK	0	W1	**Set MINTICK Interrupt Flag**
	Write 1 to set the MINTICK interrupt flag			
5	CNTTICK	0	W1	**Set CNTTICK Interrupt Flag**
	Write 1 to set the CNTTICK interrupt flag			
4	OSCFAIL	0	W1	**Set OSCFAIL Interrupt Flag**
	Write 1 to set the OSCFAIL interrupt flag			
3	CC2	0	W1	**Set CC2 Interrupt Flag**
	Write 1 to set the CC2 interrupt flag			
2	CC1	0	W1	**Set CC1 Interrupt Flag**
	Write 1 to set the CC1 interrupt flag			
1	CC0	0	W1	**Set CC0 Interrupt Flag**
	Write 1 to set the CC0 interrupt flag			
0	OF	0	W1	**Set OF Interrupt Flag**
	Write 1 to set the OF interrupt flag			

12.5.9 RTCC_IFC - Interrupt Flag Clear 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																						0	0	0	0	0	0	0	0	0	0	0
Access																						(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name																						MONTHTICK	DAYOWOF	DAYTICK	HOURTICK	MINTICK	CNTTICK	OSCFAIL	CC2	CC1	CC0	OF

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	MONTHTICK	0	(R)W1	**Clear MONTHTICK Interrupt Flag**
	Write 1 to clear the MONTHTICK interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	DAYOWOF	0	(R)W1	**Clear DAYOWOF Interrupt Flag**
	Write 1 to clear the DAYOWOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	DAYTICK	0	(R)W1	**Clear DAYTICK Interrupt Flag**
	Write 1 to clear the DAYTICK interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	HOURTICK	0	(R)W1	**Clear HOURTICK Interrupt Flag**
	Write 1 to clear the HOURTICK interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
6	MINTICK	0	(R)W1	**Clear MINTICK Interrupt Flag**
	Write 1 to clear the MINTICK interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
5	CNTTICK	0	(R)W1	**Clear CNTTICK Interrupt Flag**
	Write 1 to clear the CNTTICK interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
4	OSCFAIL	0	(R)W1	**Clear OSCFAIL Interrupt Flag**
	Write 1 to clear the OSCFAIL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	CC2	0	(R)W1	**Clear CC2 Interrupt Flag**
	Write 1 to clear the CC2 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	CC1	0	(R)W1	**Clear CC1 Interrupt Flag**
	Write 1 to clear the CC1 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	CC0	0	(R)W1	**Clear CC0 Interrupt Flag**
	Write 1 to clear the CC0 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
0	OF	0	(R)W1	**Clear OF Interrupt Flag**
	Write 1 to clear the OF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

12.5.10 RTCC_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																						0	0	0	0	0	0	0	0	0	0	0
Access																						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																						MONTHTICK	DAYOWOF	DAYTICK	HOURTICK	MINTICK	CNTTICK	OSCFAIL	CC2	CC1	CC0	OF

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	MONTHTICK	0	RW	**MONTHTICK Interrupt Enable**
	Enable/disable the MONTHTICK interrupt			
9	DAYOWOF	0	RW	**DAYOWOF Interrupt Enable**
	Enable/disable the DAYOWOF interrupt			
8	DAYTICK	0	RW	**DAYTICK Interrupt Enable**
	Enable/disable the DAYTICK interrupt			
7	HOURTICK	0	RW	**HOURTICK Interrupt Enable**
	Enable/disable the HOURTICK interrupt			
6	MINTICK	0	RW	**MINTICK Interrupt Enable**
	Enable/disable the MINTICK interrupt			
5	CNTTICK	0	RW	**CNTTICK Interrupt Enable**
	Enable/disable the CNTTICK interrupt			
4	OSCFAIL	0	RW	**OSCFAIL Interrupt Enable**
	Enable/disable the OSCFAIL interrupt			
3	CC2	0	RW	**CC2 Interrupt Enable**
	Enable/disable the CC2 interrupt			
2	CC1	0	RW	**CC1 Interrupt Enable**
	Enable/disable the CC1 interrupt			
1	CC0	0	RW	**CC0 Interrupt Enable**
	Enable/disable the CC0 interrupt			
0	OF	0	RW	**OF Interrupt Enable**
	Enable/disable the OF interrupt			

12.5.11 RTCC_STATUS - Status 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																																0
Access																																R
Name																																BUMODETS

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	BUMODETS	0	R	**Timestamp for Backup Mode Entry Stored**
	Set when a timestamp has been stored in RTCC_CC2_CCV.			

12.5.12 RTCC_CMD - Command Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																																0
Access																																W1
Name																																CLRSTATUS

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	CLRSTATUS	0	W1	**Clear RTCC_STATUS Register**
	Write a 1 to clear the RTCC_STATUS register.			

12.5.13 RTCC_SYNCBUSY - Synchronization 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																											0					
Access																											R					
Name																											CMD					

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	CMD	0	R	**CMD Register Busy**
	Set when the value written to CMD is being synchronized.			
4:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

12.5.14 RTCC_POWERDOWN - Retention RAM Power-down Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																																0
Access																																RW
Name																																RAM

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	RAM	0	RW	**Retention RAM Power-down**
	Shut off power to the Retention RAM. Once it is powered down, it cannot be powered up again			

12.5.15 RTCC_LOCK - Configuration Lock Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																	0x0000															
Access																	RWH															
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	0x0000	RWH	**Configuration Lock Key**

Write any other value than the unlock code to lock RTCC_CTRL, RTCC_PRECNT, RTCC_CNT, RTCC_TIME, RTCC_DATE, RTCC_IEN, RTCC_POWERDOWN, and RTCC_CCx_XXX registers from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.

Mode	Value	Description
Read Operation		
UNLOCKED	0	All registers are unlocked
LOCKED	1	Registers are locked
Write Operation		
LOCK	0	Lock registers
UNLOCK	0xAEE8	Unlock all RTCC registers

12.5.16 RTCC_EM4WUEN - Wake Up Enable

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																																0
Access																																RW
Name																																EM4WU

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	EM4WU	0	RW	**EM4 Wake-up Enable**

Write 1 to enable wake-up request, write 0 to disable wake-up request.

12.5.17 RTCC_CCx_CTRL - CC Channel Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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															0	0x00					0		0x0				0x0		0x0		0x0	
Access															RW	RW					RW		RW				RW		RW		RW	
Name															DAYCC	COMPMASK					COMPBASE		PRSSEL				ICEDGE		CMOA		MODE	

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	DAYCC	0	RW	**Day Capture/Compare Selection**
	Select whether day of week, or day of month is subject for Capture/Compare.			
	Value	Mode		Description
	0	MONTH		Day of month is selected for Capture/Compare.
	1	WEEK		Day of week is selected for Capture/Compare.
16:12	COMPMASK	0x00	RW	**Capture Compare Channel Comparison Mask**
	The COMPMASK most significant bits of the compare value will not be subject to comparison.			
11	COMPBASE	0	RW	**Capture Compare Channel Comparison Base**
	Configure comparison base for compare channel			
	Value	Mode		Description
	0	CNT		RTCC_CCx_CCV is compared with RTCC_CNT register. RTCC_CCx_TIME/DATE compare with RTCC_TIME/DATE in calendar mode.
	1	PRECNT		Least significant bits of RTCC_CCx_CCV are compared with PRECNT.
10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:6	PRSSEL	0x0	RW	**Compare/Capture Channel PRS Input Channel Selection**
	Select PRS input channel for Compare/Capture channel.			
	Value	Mode		Description
	0	PRSCH0		PRS Channel 0 selected as input
	1	PRSCH1		PRS Channel 1 selected as input
	2	PRSCH2		PRS Channel 2 selected as input
	3	PRSCH3		PRS Channel 3 selected as input
	4	PRSCH4		PRS Channel 4 selected as input
	5	PRSCH5		PRS Channel 5 selected as input

Bit	Name	Reset	Access	Description
	6	PRSCH6		PRS Channel 6 selected as input
	7	PRSCH7		PRS Channel 7 selected as input
	8	PRSCH8		PRS Channel 8 selected as input
	9	PRSCH9		PRS Channel 9 selected as input
	10	PRSCH10		PRS Channel 10 selected as input
	11	PRSCH11		PRS Channel 11 selected as input
	12	PRSCH12		PRS Channel 12 selected as input
	13	PRSCH13		PRS Channel 13 selected as input
	14	PRSCH14		PRS Channel 14 selected as input
	15	PRSCH15		PRS Channel 15 selected as input
5:4	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
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
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

12.5.18 RTCC_CCx_CCV - Capture/Compare Value Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x00000000																															
Access	RWH																															
Name	CCV																															

Bit	Name	Reset	Access	Description
31:0	CCV	0x00000000	RWH	**Capture/Compare Value**
	Shows the Capture/Compare Value for the channel. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = CALENDAR.			

12.5.19 RTCC_CCx_TIME - Capture/Compare Time Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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			
Access											RWH		RWH					RWH			RWH					RWH			RWH			
Name											HOURT		HOURU					MINT			MINU					SECT			SECU			

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	HOURT	0x0	RWH	**Hours, Tens**
	Shows the tens part of the Capture/Compare value for hours. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
19:16	HOURU	0x0	RWH	**Hours, Units**
	Shows the unit part of the Capture/Compare value for hours. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:12	MINT	0x0	RWH	**Minutes, Tens**
	Shows the tens part of the Capture/Compare value for minutes. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
11:8	MINU	0x0	RWH	**Minutes, Units**
	Shows the unit part of the Capture/Compare value for minutes. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	SECT	0x0	RWH	**Seconds, Tens**
	Shows the tens part of the Capture/Compare value for seconds. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
3:0	SECU	0x0	RWH	**Seconds, Units**
	Shows the unit part of the Capture/Compare value for seconds. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			

12.5.20 RTCC_CCx_DATE - Capture/Compare Date Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																				0	0x0						0x0		0x0			
Access																				RWH	RWH						RWH		RWH			
Name																				MONTHT	MONTHU						DAYT		DAYU			

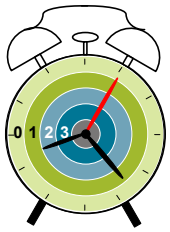
Bit	Name	Reset	Access	Description
31:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	MONTHT	0	RWH	**Month, Tens**
	Shows the tens part of the Capture/Compare value for months. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
11:8	MONTHU	0x0	RWH	**Month, Units**
	Shows the unit part of the Capture/Compare value for months. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	DAYT	0x0	RWH	**Day of Month/week, Tens**
	Shows the tens part of the Capture/Compare value for days. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			
3:0	DAYU	0x0	RWH	**Day of Month/week, Units**
	Shows the unit part of the Capture/Compare value for days. Register can not be written and will be read as zero when RTCC_CTRL_CNTMODE = NORMAL.			

12.5.21 RTCC_RETx_REG - Retention 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																0xXXXXXXXX																
Access																RW																
Name																REG																

Bit	Name	Reset	Access	Description
31:0	REG	0xXXXXXXXX	RW	**General Purpose Retention Register**

13. RTC - Real Time Counter



Quick Facts

What?

The Real Time Counter (RTC) ensures timekeeping in low energy modes. Combined with three low power oscillators (XTAL or RC), the RTC can run in EM2 DeepSleep with total current consumption less than 1.9 µA and in EM3 Stop with total current consumption less than 1.5 µA.

Why?

Timekeeping over long time periods is required in many applications, while using as little power as possible.

How?

Selectable 1000 Hz and 32768 Hz oscillators that can be used as clock source and 6 different compare registers that can trigger a wake-up. 24-bit resolution and selectable prescaling allow the system to stay in EM2 DeepSleep and EM3 Stop for a long duration of time with reliable accuracy and low power consumption.

13.1 Introduction

The Real Time Counter (RTC) contains a 24-bit counter and is clocked either by a 32768 Hz crystal oscillator, a 32768 Hz RC oscillator or a 1000 Hz RC oscillator . In addition to energy modes EM0 Active and EM1 Sleep, the RTC is also available in EM2 DeepSleep. The availability of the RTC in EM2 DeepSleep, where most of the device is powered down, makes it ideal for keeping track of time in EM2 DeepSleep. Moreover, using the 1000 Hz ULFRCO as input clock, the RTC can be used for timekeeping all the way down to EM3 Stop.

6 compare channels are available in the RTC. These can be used to trigger interrupts and to wake the device up from a low energy mode.

13.2 Features

- 24-bit Real Time Counter.
- Prescaler
  - $f_{LFACLK}/2^N$, f_{LFACLK} = 32768 Hz or 1000 Hz, N = 0 - 15.
  - Overflow @ 0.14 hours (32768 Hz) or @ 4.66 hours (1000 Hz) for prescaler setting = 0.
  - Overflow @ 194 days (32768 Hz) or @ 17.4 years (1000 Hz) for prescaler setting = 15.
- 6 compare registers
  - A compare match can potentially wake-up the device from low energy modes EM1 Sleep, EM2 DeepSleep and EM3 Stop.
  - COMP[0] compare register can be top value for RTC.
  - Overflow and Compare match events are available to other peripherals through the Peripheral Reflex System (PRS).

13.3 Functional Description

The RTC is a 24-bit counter with 6 compare channels. An overview of the RTC module is shown in Figure 13.1 RTC Overview on page 514.

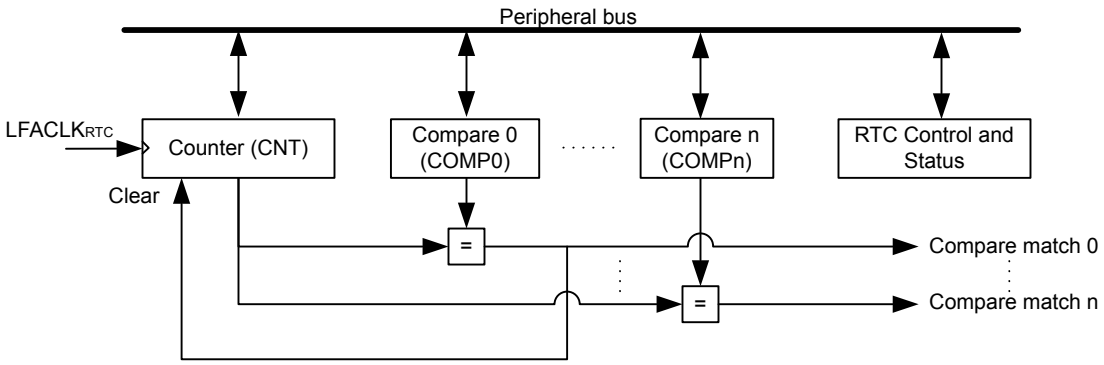


Figure 13.1. RTC Overview

13.3.1 Counter

The RTC is enabled by setting the EN bit in the RTC_CTRL register. It counts up as long as it is enabled, and will simply wrap around on an overflow and continue counting. The RTC is cleared when it is disabled. The timer value is both readable and writable and the RTC always starts counting from 0 when enabled. The value of the counter can be read or modified using the RTC_CNT register.

13.3.1.1 Clock Source

The RTC clock source and its prescaler value are defined in the Register Description section of the Clock Management Unit (CMU). The clock used by the RTC has a frequency given by Figure 13.2 RTC Frequency Equation on page 515.

$$f_{RTC} = f_{LFACLK}/2^{RTC\_PRESC}$$

Figure 13.2. RTC Frequency Equation

where f_{LFACLK} is the LFACLK frequency and RTC_PRESC is a 4 bit value. The clock source for LFACLK can be selected in CMU_LFACLKSEL_LFA and the RTC_PRESC value can be set in CMU_LFAPRESC0_RTC. Table 13.1 RTC Resolution Vs Overflow, f_{LFACLK} = 32768 Hz on page 515 shows the time of overflow and resolution of the RTC at the available prescaler values when f_{LFACLK} = 32768 Hz.

To use this module, the LE interface clock must be enabled in CMU_HFBUSCLKEN0 in addition to the module clock

Table 13.1. RTC Resolution Vs Overflow, f_{LFACLK} = 32768 Hz

RTC_PRESC	Resolution	Overflow
0	30,5 µs	512 s
1	61,0 µs	1024 s
2	122 µs	2048 s
3	244 µs	1,14 hours
4	488 µs	2,28 hours
5	977 µs	4,55 hours
6	1,95 ms	9,10 hours
7	3,91 ms	18,2 hours
8	7,81 ms	1,52 days
9	15,6 ms	3,03 days
10	31,25 ms	6,07 days
11	62,5 ms	12,1 days
12	0,125 s	24,3 days
13	0,25 s	48,5 days
14	0,5 s	97,1 days
15	1 s	194 days

13.3.2 Compare Channels

6 compare channels are available in the RTC. The compare values can be set by writing to the RTC compare channel registers RTC_COMPx, and when RTC_CNT is equal to one of these, the respective compare interrupt flag COMPn is set.

If COMP0TOP is set, the compare value set for compare channel 0 is used as a top value for the RTC, and the timer is cleared on a compare match with compare channel 0. If using the COMP0TOP setting, make sure to set this bit prior to or at the same time the EN bit is set. Setting COMP0TOP after the EN bit is set may cause unintended operation (i.e. if CNT > COMP0).

13.3.2.1 PRS Sources

All the compare channels of the RTC can be used as PRS sources. They will generate a pulse lasting one RTC clock cycle on a compare match or overflow.

13.3.3 Interrupts

The interrupts generated by the RTC are combined into one interrupt vector. If interrupts for the RTC is enabled, an interrupt will be made if one or more of the interrupt flags in RTC_IF and their corresponding bits in RTC_IEN are set. Interrupt events are overflow and compare match on either compare channels. Clearing of an interrupt flag is performed by writing to the corresponding bit in the RTC_IFC register.

13.3.4 Debugrun

By default, the RTC is halted when code execution is halted from the debugger. By setting the DEBUGRUN bit in the RTC_CTRL register, the RTC will continue to run even when the debugger is halted.

13.3.5 Using the RTC in EM3

The RTC can be enabled all the way down to EM3 Stop by using the ULFRCO as clock source. This is done by clearing CMU_LFACLKSEL_LFA and setting CMU_LFACLKSEL_LFA to ULFRCO. This will make the RTC use the internal 1000 Hz 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.

13.3.6 Register Access

This module is a Low Energy Peripheral, and supports immediate synchronization. For description regarding immediate synchronization, the reader is referred to 4.3.1 Writing.

13.4 Register Map

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

Offset	Name	Type	Description
0x000	RTC_CTRL	RW	Control Register
0x004	RTC_CNT	RWH	Counter Value Register
0x008	RTC_IF	R	Interrupt Flag Register
0x00C	RTC_IFS	W1	Interrupt Flag Set Register
0x010	RTC_IFC	(R)W1	Interrupt Flag Clear Register
0x014	RTC_IEN	RW	Interrupt Enable Register
0x020	RTC_COMPA_COMP	RW	Compare Value Register X
...	RTC_COMPx_COMP	RW	Compare Value Register X
0x034	RTC_COMPF_COMP	RW	Compare Value Register X

13.5 Register Description

13.5.1 RTC_CTRL - Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																														0	0	0
Access																														RW	RW	RW
Name																														COMP0TOP	DEBUGRUN	EN

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	COMP0TOP	0	RW	**Compare Channel 0 is Top Value**

When set, the counter is cleared in the clock cycle after a compare match with compare channel 0.

Value	Mode	Description
0	DISABLE	The top value of the RTC is 16777215 (0xFFFFFF)
1	ENABLE	The top value of the RTC is given by COMP0

Bit	Name	Reset	Access	Description
1	DEBUGRUN	0	RW	**Debug Mode Run Enable**

Set this bit to enable the RTC to keep running in debug.

Value	Description
0	RTC is frozen in debug mode
1	RTC is running in debug mode

Bit	Name	Reset	Access	Description
0	EN	0	RW	**RTC Enable**

When this bit is set, the RTC is enabled and counts up. When cleared, the counter register CNT is reset.

13.5.2 RTC_CNT - Counter Value 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									0x000000																							
Access									RWH																							
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	0x000000	RWH	**Counter Value**
	Gives access to the counter value of the RTC.			

13.5.3 RTC_IF - Interrupt Flag 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																										0x00						0
Access																										R						R
Name																										COMP						OF

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*		
6:1	COMP	0x00	R	**Compare Match X Interrupt Flag**
	Set on a compare match between CNT and COMPx.			
0	OF	0	R	**Overflow Interrupt Flag**
	Set on a CNT value overflow.			

13.5.4 RTC_IFS - Interrupt Flag Set 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																										0x00						0
Access																										W1						W1
Name																										COMP						OF

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*		
6:1	COMP	0x00	W1	**Set COMP Interrupt Flag**
	Write 1 to set the COMP interrupt flag			
0	OF	0	W1	**Set OF Interrupt Flag**
	Write 1 to set the OF interrupt flag			

13.5.5 RTC_IFC - Interrupt Flag Clear 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																										0x00						0
Access																										(R)W1						(R)W1
Name																										COMP						OF

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*		
6:1	COMP	0x00	(R)W1	**Clear COMP Interrupt Flag**
	Write 1 to clear the COMP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	OF	0	(R)W1	**Clear OF Interrupt Flag**
	Write 1 to clear the OF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

13.5.6 RTC_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																										0x00						0
Access																										RW						RW
Name																										COMP						OF

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*		
6:1	COMP	0x00	RW	**COMP Interrupt Enable**
	Enable/disable the COMP interrupt			
0	OF	0	RW	**OF Interrupt Enable**
	Enable/disable the OF interrupt			

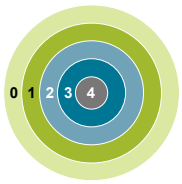
13.5.7 RTC_COMPx_COMP - Compare Value Register X (Async Reg)

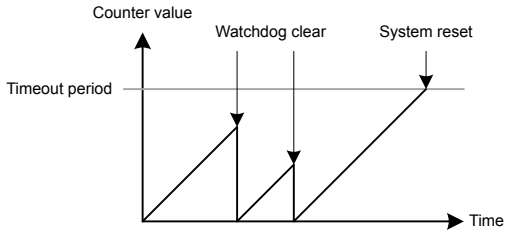
For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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									0x000000																							
Access									RW																							
Name									COMP																							

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	COMP	0x000000	RW	**Compare Value**
	A compare match event occurs when CNT is equal to this value. This event sets the COMPx interrupt flag, and can be used to start the LETIMER. It is also available as a PRS signal.			

14. WDOG - Watchdog Timer





Quick Facts

What?

The Watchdog Timer (WDOG) 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.

14.1 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.

14.2 Features

- Clock input from selectable oscillators
  - Internal 32 kHz LFRCO oscillator
  - Internal 1 kHz ULFRCO oscillator
  - External 32.768 kHz LFXO XTAL oscillator
  - HFCORECLK
- 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
- Selection to block the CMU from disabling the selected watchdog clock
- 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

14.3 Functional Description

The watchdog is enabled by setting the EN bit in WDOGn_CTRL. When enabled, the watchdog counts up to the period value configured through the PERSEL field in WDOGn_CTRL. 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 the LOCK bit in WDOGn_CTRL. Once locked, it cannot be disabled or reconfigured by software.

When the EN bit in WDOGn_CTRL is cleared to 0, the watchdog counter is reset.

14.3.1 Clock Source

Three clock sources are available for use with the watchdog, through the CLKSEL field in WDOGn_CTRL. The corresponding clocks must be enabled in the CMU. The SWOSCBLOCK bit in WDOGn_CTRL can be written to prevent accidental disabling of the selected clocks. Also, setting this bit will automatically start the selected oscillator source when the watchdog is enabled. The PERSEL field in WDOGn_CTRL is used to divide the selected watchdog clock, and the timeout for the watchdog timer can be calculated with the formula:

$$T_{TIMEOUT} = (2^{3+PERSEL} + 1) / f$$

where f is the frequency of the selected clock.

When the watchdog is enabled, it is recommended to clear the watchdog before changing PERSEL.

To use this module, the LE interface clock must be enabled in CMU_HFBUSCLKEN0.

14.3.2 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_CTRL. When code execution is resumed, the watchdog will continue counting where it left off.

14.3.3 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_CTRL. 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 Hibernate/Shutoff. If EM4BLOCK in WDOGn_CTRL is set, the CPU will be prevented from entering EM4 Hibernate/Shutoff by software request.

Note:

If the WDOG is clocked by the LFXO or LFRCO, writing the SWOSCBLOCK bit will prevent the CPU from entering EM3 Stop. When running from the ULFRCO, writing the SWOSCBLOCK bit will prevent the CPU from entering EM4 Hibernate/Shutoff.

14.3.4 Register Access

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. Refer to 4.3 Access to Low Energy Peripherals (Asynchronous Registers) for a description on how to perform register accesses to Low Energy Peripherals. Note that clearing the EN bit in WDOGn_CTRL will reset the WDOG module, which will halt any ongoing register synchronization.

Note:

Never write to the WDOG registers when it is disabled, except to enable the watchdog by setting the EN bitfield in WDOGn_CTRL.

14.3.5 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_CTRL 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:

$$T_{WARNING} = (2^{3+PERSEL}) * (WARNSEL / 4) + 1) / f,$$

where f is the frequency of the selected clock.

When the watchdog is enabled, it is recommended to clear the watchdog before changing WARNSEL.

14.3.6 Window Interrupt

This interrupt occurs when the watchdog is cleared below a certain threshold. This threshold is given by the formula:

$$T_{WARNING} = (2^{3+PERSEL}) * (WINSEL/8) + 1)/f,$$

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_CTRL. Figure 14.2 WDOG Warning, Window, and Timeout on page 523 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.

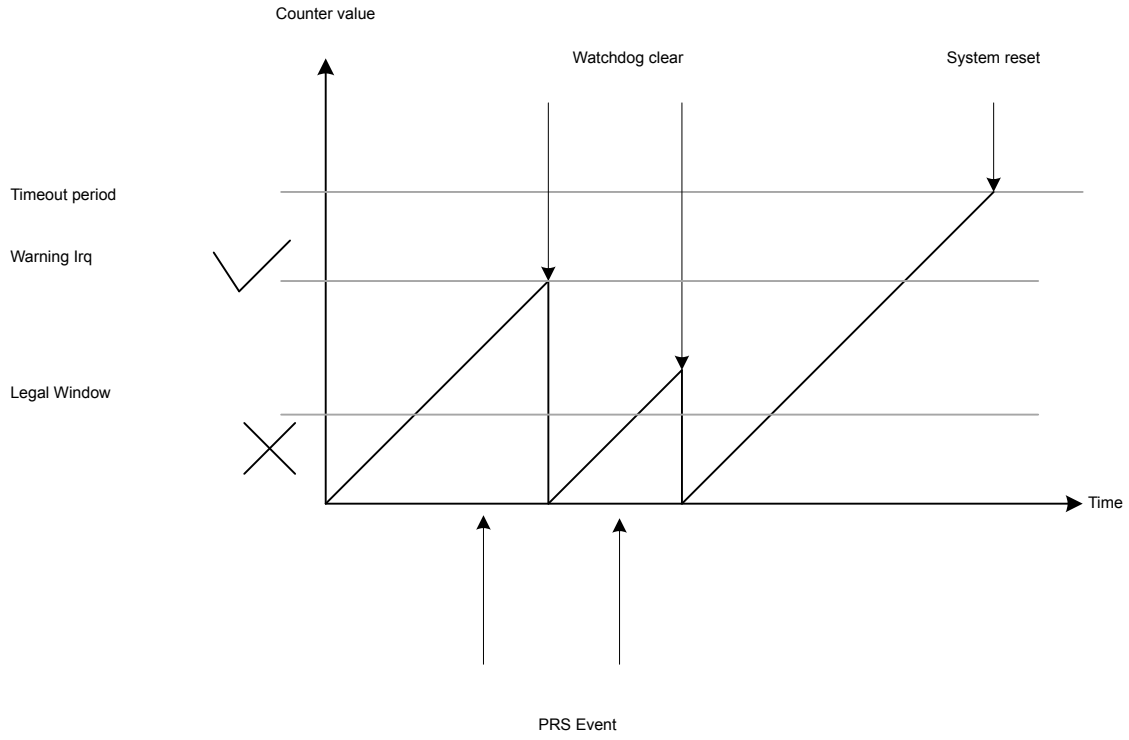


Figure 14.2. WDOG Warning, Window, and Timeout

When the watchdog is enabled, it is recommended to clear the watchdog before changing WINSEL.

14.3.7 PRS as Watchdog Clear

The first PRS channel (selected by register WDOGn_PCH0_PRSCTRL) can be used to clear the watchdog counter. To enable this feature, CLRSRC must be set to 1. Figure 14.2 PRS Clearing WDOG on page 524 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.

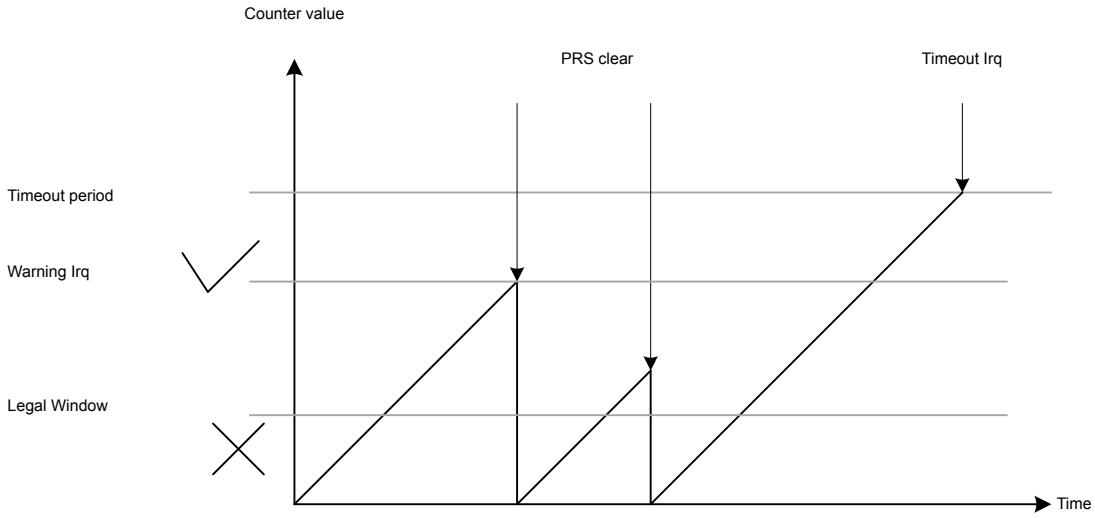


Figure 14.2. PRS Clearing WDOG

14.3.8 PRS Rising Edge Monitoring

PRS channels can be used to monitor multiple processes. 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.

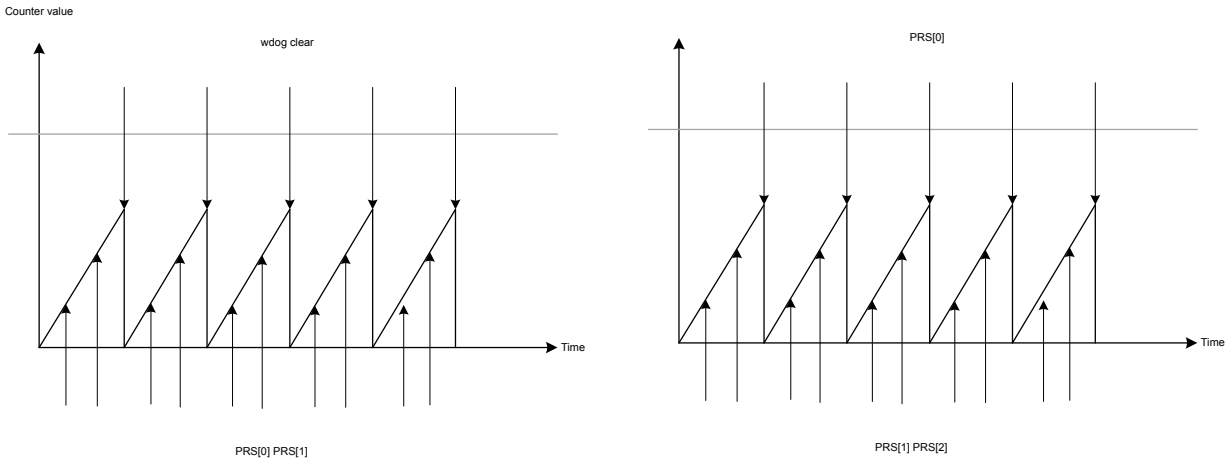


Figure 14.3. PRS Edge Monitoring in WDOG

14.4 Register Map

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

Offset	Name	Type	Description
0x000	WDOG_CTRL	RW	Control Register
0x004	WDOG_CMD	W1	Command Register
0x008	WDOG_SYNCBUSY	R	Synchronization Busy Register
0x00C	WDOGn_PCH0_PRSCTRL	RW	PRS Control Register
0x010	WDOGn_PCH1_PRSCTRL	RW	PRS Control Register
0x01C	WDOG_IF	R	Watchdog Interrupt Flags
0x020	WDOG_IFS	W1	Interrupt Flag Set Register
0x024	WDOG_IFC	(R)W1	Interrupt Flag Clear Register
0x028	WDOG_IEN	RW	Interrupt Enable Register

14.5 Register Description

14.5.1 WDOG_CTRL - Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0	0				0x0									0x0				0x0		0xF					0	0	0	0	0	0	0
Access	RW	RW				RW									RW				RW		RW					RW	RW	RW	RW	RW	RW	RW
Name	WDOGRSTDIS	CLRSRC				WINSEL									WARNSEL				CLKSEL		PERSEL					SWOSCBLOCK	EM4BLOCK	LOCK	EM3RUN	EM2RUN	DEBUGRUN	EN

Bit	Name	Reset	Access	Description
31	WDOGRSTDIS	0	RW	**Watchdog Reset Disable**
	Disable watchdog reset output.			

Value	Mode	Description
0	EN	A timeout will cause a watchdog reset
1	DIS	A timeout will not cause a watchdog reset

Bit	Name	Reset	Access	Description
30	CLRSRC	0	RW	**Watchdog Clear Source**
	Select watchdog clear source.			

Value	Mode	Description
0	SW	A write to the clear bit will clear the watchdog counter
1	PCH0	A rising edge on the PRS Channel0 will clear the watchdog counter

Bit	Name	Reset	Access	Description
29:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:24	WINSEL	0x0	RW	**Watchdog Illegal Window Select**
	Select watchdog illegal limit.			

Value	Description
0	Disabled.
1	Window limit is 12.5% of the Timeout.
2	Window limit is 25.0% of the Timeout.
3	Window limit is 37.5% of the Timeout.
4	Window limit is 50.0% of the Timeout.
5	Window limit is 62.5% of the Timeout.
6	Window limit is 75.0% of the Timeout.
7	Window limit is 87.5% of the Timeout.

Bit	Name	Reset	Access	Description
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	WARNSEL	0x0	RW	**Watchdog Timeout Period Select**

Select watchdog warning timeout period.

Value	Description
0	Disabled.
1	Warning timeout is 25% of the Timeout.
2	Warning timeout is 50% of the Timeout.
3	Warning timeout is 75% of the Timeout.

15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	CLKSEL	0x0	RW	**Watchdog Clock Select**

Selects the WDOG oscillator, i.e. the clock on which the watchdog will run.

Value	Mode	Description
0	ULFRCO	ULFRCO
1	LFRCO	LFRCO
2	LFXO	LFXO
3	HFCORECLK	HFCORECLK

11:8	PERSEL	0xF	RW	**Watchdog Timeout Period Select**

Select watchdog timeout period.

Value	Description
0	Timeout period of 9 watchdog clock cycles.
1	Timeout period of 17 watchdog clock cycles.
2	Timeout period of 33 watchdog clock cycles.
3	Timeout period of 65 watchdog clock cycles.
4	Timeout period of 129 watchdog clock cycles.
5	Timeout period of 257 watchdog clock cycles.
6	Timeout period of 513 watchdog clock cycles.
7	Timeout period of 1k watchdog clock cycles.
8	Timeout period of 2k watchdog clock cycles.
9	Timeout period of 4k watchdog clock cycles.
10	Timeout period of 8k watchdog clock cycles.
11	Timeout period of 16k watchdog clock cycles.
12	Timeout period of 32k watchdog clock cycles.
13	Timeout period of 64k watchdog clock cycles.
14	Timeout period of 128k watchdog clock cycles.

Bit	Name	Reset	Access	Description
	15			Timeout period of 256k watchdog clock cycles.
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	SWOSCBLOCK	0	RW	**Software Oscillator Disable Block**

Set to disallow disabling of the selected WDOG oscillator. Writing this bit to 1 will turn on the selected WDOG oscillator if it is not already running.

Value	Description
0	Software is allowed to disable the selected WDOG oscillator. See CMU for detailed description. Note that also CMU registers are lockable.
1	Software is not allowed to disable the selected WDOG oscillator.

5	EM4BLOCK	0	RW	**Energy Mode 4 Block**

Set to disallow EM4 entry by software.

Value	Description
0	EM4 can be entered by software. See EMU for detailed description.
1	EM4 cannot be entered by software.

4	LOCK	0	RW	**Configuration Lock**

Set to lock the watchdog configuration. This bit can only be cleared by reset.

Value	Description
0	Watchdog configuration can be changed.
1	Watchdog configuration cannot be changed.

3	EM3RUN	0	RW	**Energy Mode 3 Run Enable**

Set to keep watchdog running in EM3.

Value	Description
0	Watchdog timer is frozen in EM3.
1	Watchdog timer is running in EM3.

2	EM2RUN	0	RW	**Energy Mode 2 Run Enable**

Set to keep watchdog running in EM2.

Value	Description
0	Watchdog timer is frozen in EM2.
1	Watchdog timer is running in EM2.

1	DEBUGRUN	0	RW	**Debug Mode Run Enable**

Set to keep watchdog running in debug mode.

Value	Description
0	Watchdog timer is frozen in debug mode.
1	Watchdog timer is running in debug mode.

Bit	Name	Reset	Access	Description
0	EN	0	RW	**Watchdog Timer Enable**
	Set to enabled watchdog timer.			

14.5.2 WDOG_CMD - Command Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																																0
Access																																W1
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	0	W1	**Watchdog Timer Clear**
	Clear watchdog timer. The bit must be written 4 watchdog cycles before the timeout.			

Value	Mode	Description
0	UNCHANGED	Watchdog timer is unchanged.
1	CLEARED	Watchdog timer is cleared to 0.

14.5.3 WDOG_SYNCBUSY - Synchronization Busy 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																													0	0	0	0
Access																													R	R	R	R
Name																													PCH1_PRSCTRL	PCH0_PRSCTRL	CMD	CTRL

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	PCH1_PRSCTRL	0	R	**PCH1_PRSCTRL Register Busy**
	Set when the value written to PCH1_PRSCTRL is being synchronized.			
2	PCH0_PRSCTRL	0	R	**PCH0_PRSCTRL Register Busy**
	Set when the value written to PCH0_PRSCTRL is being synchronized.			
1	CMD	0	R	**CMD Register Busy**
	Set when the value written to CMD is being synchronized.			
0	CTRL	0	R	**CTRL Register Busy**
	Set when the value written to CTRL is being synchronized.			

14.5.4 WDOGn_PCHx_PRSCTRL - PRS Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																								0					0x0			
Access																								RW					RW			
Name																								PRSMISSRSTEN					PRSSEL			

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	PRSMISSRSTEN	0	RW	**PRS Missing Event Will Trigger a Watchdog Reset**
	When set, a PRS missing event will trigger a watchdog reset.			
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	**PRS Channel PRS Select**
	These bits select the PRS input for the PRS channel.			

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input
8	PRSCH8	PRS Channel 8 selected as input
9	PRSCH9	PRS Channel 9 selected as input
10	PRSCH10	PRS Channel 10 selected as input
11	PRSCH11	PRS Channel 11 selected as input
12	PRSCH12	PRS Channel 12 selected as input
13	PRSCH13	PRS Channel 13 selected as input
14	PRSCH14	PRS Channel 14 selected as input
15	PRSCH15	PRS Channel 15 selected as input

14.5.5 WDOG_IF - Watchdog Interrupt Flags

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																												0	0	0	0	0
Access																												R	R	R	R	R
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	0	R	**PRS Channel One Event Missing Interrupt Flag**
	Set when a WDOG clear happens before a prs event has been detected on PRS channel one.			
3	PEM0	0	R	**PRS Channel Zero Event Missing Interrupt Flag**
	Set when a WDOG clear happens before a prs event has been detected on PRS channel zero.			
2	WIN	0	R	**WDOG Window Interrupt Flag**
	Set when a WDOG clear happens below the window limit value.			
1	WARN	0	R	**WDOG Warning Timeout Interrupt Flag**
	Set when a WDOG warning timeout has occurred.			
0	TOUT	0	R	**WDOG Timeout Interrupt Flag**
	Set when a WDOG timeout has occurred.			

14.5.6 WDOG_IFS - Interrupt Flag Set 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																												0	0	0	0	0
Access																												W1	W1	W1	W1	W1
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	0	W1	**Set PEM1 Interrupt Flag**
	Write 1 to set the PEM1 interrupt flag			
3	PEM0	0	W1	**Set PEM0 Interrupt Flag**
	Write 1 to set the PEM0 interrupt flag			
2	WIN	0	W1	**Set WIN Interrupt Flag**
	Write 1 to set the WIN interrupt flag			
1	WARN	0	W1	**Set WARN Interrupt Flag**
	Write 1 to set the WARN interrupt flag			
0	TOUT	0	W1	**Set TOUT Interrupt Flag**
	Write 1 to set the TOUT interrupt flag			

14.5.7 WDOG_IFC - Interrupt Flag Clear 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																												0	0	0	0	0
Access																												(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name																												PEM1	PEM0	WIN	WARN	TOUT

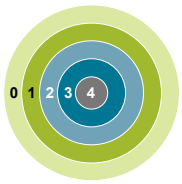
Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	PEM1	0	(R)W1	**Clear PEM1 Interrupt Flag**
	Write 1 to clear the PEM1 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	PEM0	0	(R)W1	**Clear PEM0 Interrupt Flag**
	Write 1 to clear the PEM0 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	WIN	0	(R)W1	**Clear WIN Interrupt Flag**
	Write 1 to clear the WIN interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	WARN	0	(R)W1	**Clear WARN Interrupt Flag**
	Write 1 to clear the WARN interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	TOUT	0	(R)W1	**Clear TOUT Interrupt Flag**
	Write 1 to clear the TOUT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

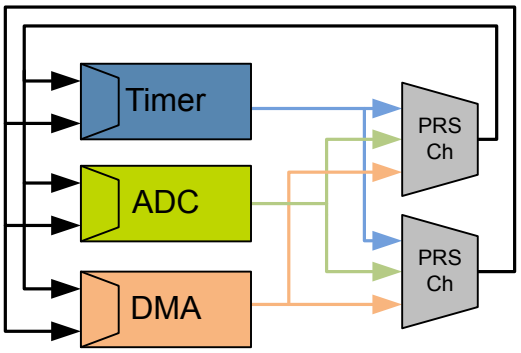
14.5.8 WDOG_IEN - Interrupt Enable Register

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																												0	0	0	0	0
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	0	RW	**PEM1 Interrupt Enable**
	Enable/disable the PEM1 interrupt			
3	PEM0	0	RW	**PEM0 Interrupt Enable**
	Enable/disable the PEM0 interrupt			
2	WIN	0	RW	**WIN Interrupt Enable**
	Enable/disable the WIN interrupt			
1	WARN	0	RW	**WARN Interrupt Enable**
	Enable/disable the WARN interrupt			
0	TOUT	0	RW	**TOUT Interrupt Enable**
	Enable/disable the TOUT interrupt			

15. PRS - Peripheral Reflex System





Quick Facts

What?

The Peripheral Reflex System (PRS) allows configurable, fast, and autonomous communication between peripherals.

Why?

Events and signals from one peripheral can be used as input signals or triggered by other peripherals. Besides, 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 (both pulses and level) 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.

15.1 Introduction

The Peripheral Reflex System (PRS) is a network allowing direct communication between different peripheral modules without involving the CPU. Peripheral modules which send out Reflex signals are called producers. The PRS routes these Reflex signals through Reflex channels to consumer peripherals which perform actions depending on the Reflex signals received. The format for the Reflex signals is not given, but edge triggers and other functionality can be applied by the PRS.

15.2 Features

- 16 Configurable Reflex Channels
  - Each channel can be connected to any producing peripheral, including the PRS channels
  - Consumers can choose which channel to listen to
  - Selectable edge detector (Rising, falling and both edges)
  - Configurable AND and OR between channels
  - Optional channel invert
  - PRS can generate event to CPU
  - Two independent DMA requests based on PRS channels
- Software controlled channel output
  - Configurable level
  - Triggered pulses

15.3 Functional Description

An overview of the PRS module is shown in Figure 15.1 PRS Overview on page 537. The PRS contains 16 Reflex channels. All channels can select any Reflex signal offered by the producers. The consumers can choose which PRS channel to listen to and perform actions based on the Reflex signals routed through that channel. The Reflex signals can be both edge signals and level signals.

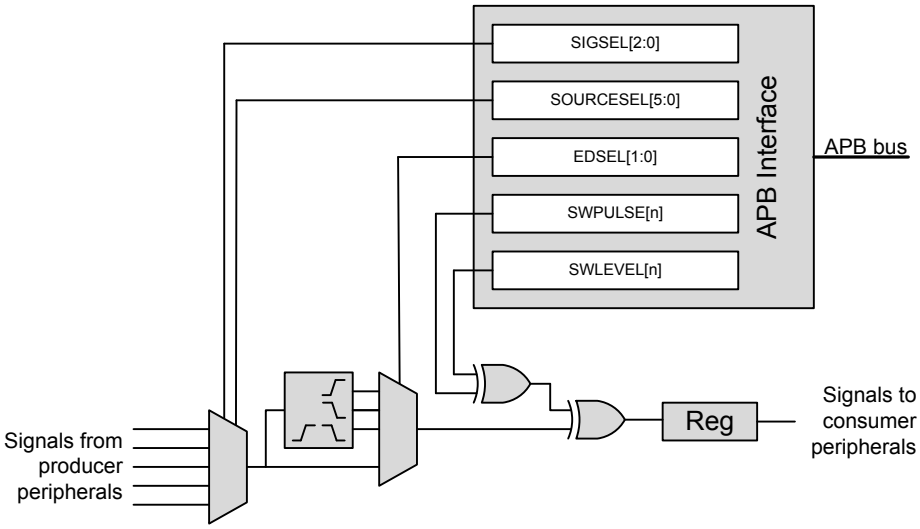


Figure 15.1. PRS Overview

15.3.1 Channel Functions

Different functions can be applied to a Reflex signal within the PRS. Each channel includes an edge detector to enable generation of pulse signals from level signals. The PRS channels can also be manually triggered by writing to PRS_SWPULSE or PRS_SWLEVEL. SWLEVEL[n] is a programmable level for each channel and holds the value it is programmed to. Setting SWPULSE[n] will cause the PRS channel to output a high pulse that is one HFCLK cycle wide. The SWLEVEL[n] and SWPULSE[n] signals are then XOR'ed with the selected input from the producers to form the output signal sent to the consumers listening to the channel. For example, when SWLEVEL[n] is set, if a producer produces a signal of 1, this will cause a channel output of 0.

15.3.1.1 Operational Mode

Reflex channels can operate in two modes, synchronous or asynchronous. In synchronous mode Reflex signals are clocked on the HFCLK, and can be used by any Reflex consumer. However, this will not work in EM2/EM3, since the HFCLK will be turned off.

Asynchronous Reflex channels are not clocked on HFCLK, and can be used even in EM2/EM3. However, the asynchronous mode can only be used by a subset of the Reflex consumers.

The asynchronous Reflex signals generated by the producers are indicated in the SIGSEL field of PRS_CHx_CTRL register. The consumers capable of utilizing asynchronous Reflex signals include the LEUART and the PCNT. The USART can also utilize some particular asynchronous signals. Refer to the respective modules for details on how to configure them to use the PRS.

Note: If a Reflex channel with ASYNC field of PRS_CHx_CTRL register set to '1' is used in a consumer not supporting asynchronous reflexes, the behaviour is undefined

15.3.1.2 Edge Detection and Clock Domains

Using EDSEL in PRS_CHx_CTRL, edge detection can be applied to a PRS signal. When edge detection is enabled, changes in the PRS input will result in a pulse on the PRS channel. This requires that the ASYNC bit in PRS_CHx_CTRL is cleared. Signals on the PRS input must be at least one HFCLK period wide in order to be detected properly. This applies to all cases when ASYNC is not used in the PRS.

For communication between peripherals on different prescaled clocks (e.g. between peripherals on HFCLK and HFPERCLK), there are two options. One option is to use level signals. No additional action is needed for level signals, but software must make sure that the level signals are held long enough for the destination domain to detect them. The other option is to use pulse signals. For pulse signals, edge detection should be enabled (by configuring EDSEL in PRS_CHx_CTRL to positive edge, negative edge, or both) and STRETCH in PRS_CHx_CTRL should be set. When edge detection and stretch are enabled on a PRS source, the output on the PRS channel is held long enough for the destination domain to detect the pulse. This also works if there are multiple destination domains running at different frequencies.

15.3.1.3 Configurable PRS Logic

Each PRS channel has three logic functions that can be used by themselves or in combination. The selected PRS source can be AND'ed with the next PRS channel output, OR'ed with the previous PRS channel output and inverted. This is shown in Figure 15.1 PRS Overview on page 537. The order of the functions is important. If OR and AND are enabled at the same time, AND is applied first, and then OR. Note that the previous and next channel options wrap around. Using the ORPREV option on the first PRS channel OR's with the output of the last PRS channel. Likewise, using the ANDNEXT option on the last PRS channel AND's with the output of the first PRS channel.

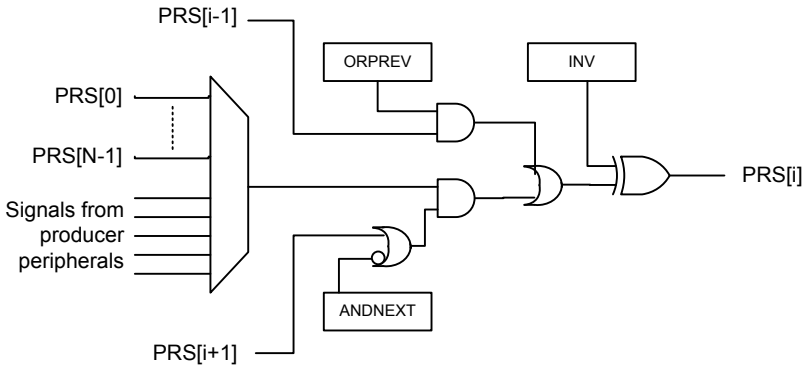


Figure 15.2. Configurable PRS Logic

In addition to the logic functions that can combine a PRS channel with one of its neighbors, a PRS channel can also select any other PRS channel as input. This can allow relatively complex logic functions to be created.

15.3.2 Producers

Through SOURCESEL in PRS_CHx_CTRL, each PRS channel selects signal producers. Each producer outputs one or more signals which can be selected by setting the SIGSEL field in PRS_CHx_CTRL. Setting the SOURCESEL bits to 0 (Off) leads to a constant 0 output from the input mux. An overview of the available producers can be found in the SOURCESEL and SIGSEL fields in PRS_CHx_CTRL. Note that GPIO producers are selected in the GPIO module using the edge interrupt configuration settings described in 34.3.5.1 Edge Interrupt Generation. GPIOPIN0 uses the selection for the EXTI0 interrupt, GPIOPIN1 uses the selection for the EXTI1 interrupt, and so on.

15.3.3 Consumers

Consumer peripherals (Listed in Table 15.1 Reflex Consumers on page 539) can be set to listen to a PRS channel and perform an action based on the signal received on that channel. While most consumers can handle either only pulse input or only level input, some can handle both pulse and level inputs.

Table 15.1. Reflex Consumers

Module	Reflex Input	Input Format
TIMER	Compare/Capture Channel	Pulse / Level
	Alternate Input for DTI (Available only in specific TIMERs See data sheet for details)	Level
	Alternate Input for DTI Fault 0 (Available only in specific TIMERs See data sheet for details)	Level
	Alternate Input for DTI Fault 1 (Available only in specific TIMERs See data sheet for details)	Level
WTIMER	Compare/Capture Channel	Pulse / Level
	Alternate Input for DTI (Available only in specific WTIMERs See data sheet for details)	Level
	Alternate Input for DTI Fault 0 (Available only in specific WTIMERs See data sheet for details)	Level
	Alternate Input for DTI Fault 1 (Available only in specific WTIMERs See data sheet for details)	Level
USART	RX/TX Trigger	Pulse
	Alternate Input for IrDA	Level
	Alternate Input for RX	Level
	Alternate Input for CLK	Level
VDAC	Channel 0 Trigger	Pulse
	Channel 1 Trigger	Pulse
ADC	Single Sample Trigger	Pulse
	Scan Sequence Trigger	Pulse
IDAC	Alternate Input for OUTMODE	Level
CMU	Alternate Input for Calibration Up-Counter	Level
	Alternate Input for Calibration Down-Counter	Level
LEUART	Alternate Input for RX	Level
PCNT	Compare/Clear Trigger	Pulse/Level
	Alternate Input for S0IN	Level
	Alternate Input for S1IN	Level

Module	Reflex Input	Input Format
LESENSE	Scan Start	Pulse
	LESENSE Decoder Bit 0	Level
	LESENSE Decoder Bit 1	Level
	LESENSE Decoder Bit 2	Level
	LESENSE Decoder Bit 3	Level
WDOG	Peripheral Watchdog	Pulse
LETIMER	Start LETIMER	Pulse
	Stop LETIMER	Pulse
	Clear LETIMER	Pulse
RTCC	Compare/Capture Channel	Pulse/Level
PRS	Set Event	Pulse
	DMA Request 0	Pulse
	DMA Request 1	Pulse
CAPSENSE	Start Conversion	Pulse

15.3.4 Event on PRS

The PRS can be used to send events to the MCU. This is very useful in combination with the Wait For Event (WFE) instruction. A single PRS channel can be selected for this using SEVONPRSSEL in PRS_CTRL, and the feature is enabled by setting SEVONPRS in the same register.

Using SEVONPRS, one can e.g. set up a timer to trigger an event to the MCU periodically, every time letting the MCU pass through 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, then execute some code, then wait for an event, then execute some code and so on.

15.3.5 DMA Request on PRS

Up to two independent DMA requests can be generated by the PRS. The PRS signals triggering the DMA requests are selected using the LDMA_CHx_REQSEL register, by setting SOURCESEL to PRS and SIGSEL to either PRSREQ0 or PRSREQ1. The DMA requests are cleared when the DMA services the requests. The requests are set whenever the selected PRS signals are high.

The selected PRS signals must have ASYNC cleared when they are used as inputs to the DMA. Edge detection in the PRS can be enabled to only trigger transfers on edges.

15.3.6 Example

The example below (illustrated in Figure 15.3 TIMER0 Overflow Starting ADC0 Single Conversions Through PRS Channel 5. on page 541) shows how to set up ADC0 to start single conversions every time TIMER0 overflows (one HFPERCLK cycle high pulse), using PRS channel 5:

- Set SOURCESEL in PRS_CH5_CTRL to TIMER0 as input to PRS channel 5.
- Set SIGSEL in PRS_CH5_CTRL to select the overflow signal (TIMER0OF from TIMER0).
- Configure ADC0 with the desired conversion set-up.
- Set SINGLEPRSEN in ADC0_SINGLECTRL to 1 to enable single conversions to be started by a high PRS input signal.
- Set SINGLEPRSSEL in ADC0_SINGLECTRL to 0x5 to select PRS channel 5 as input to start the single conversion.
- Start TIMER0 with the desired TOP value, an overflow PRS signal is output automatically on overflow.

Note that the ADC results needs to be fetched either by the CPU or DMA.

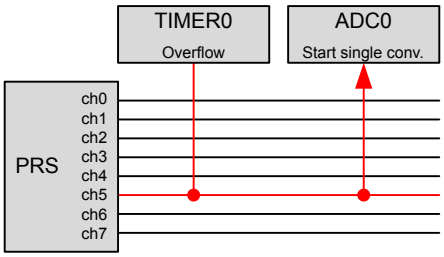


Figure 15.3. TIMER0 Overflow Starting ADC0 Single Conversions Through PRS Channel 5.

15.4 Register Map

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

Offset	Name	Type	Description
0x000	PRS_SWPULSE	W1	Software Pulse Register
0x004	PRS_SWLEVEL	RW	Software Level Register
0x008	PRS_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x010	PRS_ROUTELOC0	RW	I/O Routing Location Register
0x014	PRS_ROUTELOC1	RW	I/O Routing Location Register
0x018	PRS_ROUTELOC2	RW	I/O Routing Location Register
0x01C	PRS_ROUTELOC3	RW	I/O Routing Location Register
0x030	PRS_CTRL	RW	Control Register
0x034	PRS_DMAREQ0	RW	DMA Request 0 Register
0x038	PRS_DMAREQ1	RW	DMA Request 1 Register
0x040	PRS_PEEK	R	PRS Channel Values
0x050	PRS_CH0_CTRL	RW	Channel Control Register
...	PRS_CHx_CTRL	RW	Channel Control Register
0x08C	PRS_CH15_CTRL	RW	Channel Control Register

15.5 Register Description

15.5.1 PRS_SWPULSE - Software Pulse 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																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access																	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																	CH15PULSE	CH14PULSE	CH13PULSE	CH12PULSE	CH11PULSE	CH10PULSE	CH9PULSE	CH8PULSE	CH7PULSE	CH6PULSE	CH5PULSE	CH4PULSE	CH3PULSE	CH2PULSE	CH1PULSE	CH0PULSE

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	CH15PULSE	0	W1	**Channel 15 Pulse Generation**
	See bit 0.			
14	CH14PULSE	0	W1	**Channel 14 Pulse Generation**
	See bit 0.			
13	CH13PULSE	0	W1	**Channel 13 Pulse Generation**
	See bit 0.			
12	CH12PULSE	0	W1	**Channel 12 Pulse Generation**
	See bit 0.			
11	CH11PULSE	0	W1	**Channel 11 Pulse Generation**
	See bit 0.			
10	CH10PULSE	0	W1	**Channel 10 Pulse Generation**
	See bit 0.			
9	CH9PULSE	0	W1	**Channel 9 Pulse Generation**
	See bit 0.			
8	CH8PULSE	0	W1	**Channel 8 Pulse Generation**
	See bit 0.			
7	CH7PULSE	0	W1	**Channel 7 Pulse Generation**
	See bit 0.			
6	CH6PULSE	0	W1	**Channel 6 Pulse Generation**
	See bit 0.			
5	CH5PULSE	0	W1	**Channel 5 Pulse Generation**
	See bit 0.			
4	CH4PULSE	0	W1	**Channel 4 Pulse Generation**
	See bit 0.			

Bit	Name	Reset	Access	Description
3	CH3PULSE	0	W1	**Channel 3 Pulse Generation**
	See bit 0.			
2	CH2PULSE	0	W1	**Channel 2 Pulse Generation**
	See bit 0.			
1	CH1PULSE	0	W1	**Channel 1 Pulse Generation**
	See bit 0.			
0	CH0PULSE	0	W1	**Channel 0 Pulse Generation**
	Write to 1 to generate one HFCLK cycle high pulse. This pulse is XOR'ed with the corresponding bit in the SWLEVEL register and the selected PRS input signal to generate the channel output.			

15.5.2 PRS_SWLEVEL - Software 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																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access																	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																	CH15LEVEL	CH14LEVEL	CH13LEVEL	CH12LEVEL	CH11LEVEL	CH10LEVEL	CH9LEVEL	CH8LEVEL	CH7LEVEL	CH6LEVEL	CH5LEVEL	CH4LEVEL	CH3LEVEL	CH2LEVEL	CH1LEVEL	CH0LEVEL

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	CH15LEVEL	0	RW	**Channel 15 Software Level**
	See bit 0.			
14	CH14LEVEL	0	RW	**Channel 14 Software Level**
	See bit 0.			
13	CH13LEVEL	0	RW	**Channel 13 Software Level**
	See bit 0.			
12	CH12LEVEL	0	RW	**Channel 12 Software Level**
	See bit 0.			
11	CH11LEVEL	0	RW	**Channel 11 Software Level**
	See bit 0.			
10	CH10LEVEL	0	RW	**Channel 10 Software Level**
	See bit 0.			
9	CH9LEVEL	0	RW	**Channel 9 Software Level**
	See bit 0.			
8	CH8LEVEL	0	RW	**Channel 8 Software Level**
	See bit 0.			
7	CH7LEVEL	0	RW	**Channel 7 Software Level**
	See bit 0.			
6	CH6LEVEL	0	RW	**Channel 6 Software Level**
	See bit 0.			
5	CH5LEVEL	0	RW	**Channel 5 Software Level**
	See bit 0.			
4	CH4LEVEL	0	RW	**Channel 4 Software Level**
	See bit 0.			
3	CH3LEVEL	0	RW	**Channel 3 Software Level**
	See bit 0.			

Bit	Name	Reset	Access	Description
2	CH2LEVEL	0	RW	**Channel 2 Software Level**
	See bit 0.			
1	CH1LEVEL	0	RW	**Channel 1 Software Level**
	See bit 0.			
0	CH0LEVEL	0	RW	**Channel 0 Software Level**
	The value in this register is XOR'ed with the corresponding bit in the SWPULSE register and the selected PRS input signal to generate the channel output.			

15.5.3 PRS_ROUTEPEN - I/O Routing Pin Enable 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																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access																	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																	CH15PEN	CH14PEN	CH13PEN	CH12PEN	CH11PEN	CH10PEN	CH9PEN	CH8PEN	CH7PEN	CH6PEN	CH5PEN	CH4PEN	CH3PEN	CH2PEN	CH1PEN	CH0PEN

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	CH15PEN	0	RW	**CH15 Pin Enable**
	When set, GPIO output from PRS channel 15 is enabled			
14	CH14PEN	0	RW	**CH14 Pin Enable**
	When set, GPIO output from PRS channel 14 is enabled			
13	CH13PEN	0	RW	**CH13 Pin Enable**
	When set, GPIO output from PRS channel 13 is enabled			
12	CH12PEN	0	RW	**CH12 Pin Enable**
	When set, GPIO output from PRS channel 12 is enabled			
11	CH11PEN	0	RW	**CH11 Pin Enable**
	When set, GPIO output from PRS channel 11 is enabled			
10	CH10PEN	0	RW	**CH10 Pin Enable**
	When set, GPIO output from PRS channel 10 is enabled			
9	CH9PEN	0	RW	**CH9 Pin Enable**
	When set, GPIO output from PRS channel 9 is enabled			
8	CH8PEN	0	RW	**CH8 Pin Enable**
	When set, GPIO output from PRS channel 8 is enabled			
7	CH7PEN	0	RW	**CH7 Pin Enable**
	When set, GPIO output from PRS channel 7 is enabled			
6	CH6PEN	0	RW	**CH6 Pin Enable**
	When set, GPIO output from PRS channel 6 is enabled			
5	CH5PEN	0	RW	**CH5 Pin Enable**
	When set, GPIO output from PRS channel 5 is enabled			
4	CH4PEN	0	RW	**CH4 Pin Enable**
	When set, GPIO output from PRS channel 4 is enabled			
3	CH3PEN	0	RW	**CH3 Pin Enable**
	When set, GPIO output from PRS channel 3 is enabled			

Bit	Name	Reset	Access	Description
2	CH2PEN	0	RW	**CH2 Pin Enable**
	When set, GPIO output from PRS channel 2 is enabled			
1	CH1PEN	0	RW	**CH1 Pin Enable**
	When set, GPIO output from PRS channel 1 is enabled			
0	CH0PEN	0	RW	**CH0 Pin Enable**
	When set, GPIO output from PRS channel 0 is enabled			

15.5.4 PRS_ROUTELOC0 - I/O Routing Location 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			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			CH3LOC								CH2LOC								CH1LOC								CH0LOC					

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	CH3LOC	0x00	RW	**I/O Location**

Decides the location of the channel I/O pin

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 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*		
21:16	CH2LOC	0x00	RW	**I/O Location**

Decides the location of the channel I/O pin

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

Bit	Name	Reset	Access	Description
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CH1LOC	0x00	RW	**I/O Location**

Decides the location of the channel I/O pin

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

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*		
5:0	CH0LOC	0x00	RW	**I/O Location**

Decides the location of the channel I/O pin

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

15.5.5 PRS_ROUTELOC1 - I/O Routing Location 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			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			CH7LOC								CH6LOC								CH5LOC								CH4LOC					

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	CH7LOC	0x00	RW	**I/O Location**

Decides the location of the channel I/O pin

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

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*		
21:16	CH6LOC	0x00	RW	**I/O Location**

Decides the location of the channel I/O pin

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

Bit	Name	Reset	Access	Description
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CH5LOC	0x00	RW	**I/O Location**

Decides the location of the channel I/O pin

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

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*		
5:0	CH4LOC	0x00	RW	**I/O Location**

Decides the location of the channel I/O pin

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2

15.5.6 PRS_ROUTELOC2 - I/O Routing Location 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			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			CH11LOC								CH10LOC								CH9LOC								CH8LOC					

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	CH11LOC	0x00	RW	**I/O Location**
	Decides the location of the channel I/O pin			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

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*		
21:16	CH10LOC	0x00	RW	**I/O Location**
	Decides the location of the channel I/O pin			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

Bit	Name	Reset	Access	Description
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CH9LOC	0x00	RW	**I/O Location**
	Decides the location of the channel I/O pin			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

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*		
5:0	CH8LOC	0x00	RW	**I/O Location**
	Decides the location of the channel I/O pin			

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2

15.5.7 PRS_ROUTELOC3 - I/O Routing Location 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			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			CH15LOC								CH14LOC								CH13LOC								CH12LOC					

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	CH15LOC	0x00	RW	**I/O Location**
	Decides the location of the channel I/O pin			
	Value	Mode		Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:16	CH14LOC	0x00	RW	**I/O Location**
	Decides the location of the channel I/O pin			
	Value	Mode		Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CH13LOC	0x00	RW	**I/O Location**
	Decides the location of the channel I/O pin			
	Value	Mode		Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	CH12LOC	0x00	RW	**I/O Location**
	Decides the location of the channel I/O pin			

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2

15.5.8 PRS_CTRL - 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				0
Access																												RW				RW
Name																												SEVONPRSSEL				SEVONPRS

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*		
4:1	SEVONPRSSEL	0x0	RW	**SEVONPRS PRS Channel Select**

Selects PRS channel for SEVONPRS

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected
8	PRSCH8	PRS Channel 8 selected
9	PRSCH9	PRS Channel 9 selected
10	PRSCH10	PRS Channel 10 selected
11	PRSCH11	PRS Channel 11 selected
12	PRSCH12	PRS Channel 12 selected
13	PRSCH13	PRS Channel 13 selected
14	PRSCH14	PRS Channel 14 selected
15	PRSCH15	PRS Channel 15 selected

0	SEVONPRS	0	RW	**Set Event on PRS**

When set, an event is generated to the CPU when the PRS channel selected by SEVONPRSSEL is high

15.5.9 PRS_DMAREQ0 - DMA Request 0 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																							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*		
9:6	PRSSEL	0x0	RW	**DMA Request 0 PRS Channel Select**

Selects PRS channel for DMA request 0 from the PRS (PRSREQ0).

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected
8	PRSCH8	PRS Channel 8 selected
9	PRSCH9	PRS Channel 9 selected
10	PRSCH10	PRS Channel 10 selected
11	PRSCH11	PRS Channel 11 selected
12	PRSCH12	PRS Channel 12 selected
13	PRSCH13	PRS Channel 13 selected
14	PRSCH14	PRS Channel 14 selected
15	PRSCH15	PRS Channel 15 selected

5:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

15.5.10 PRS_DMAREQ1 - DMA Request 1 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																							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*		
9:6	PRSSEL	0x0	RW	**DMA Request 1 PRS Channel Select**

Selects PRS channel for DMA request 1 from the PRS (PRSREQ1).

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected
8	PRSCH8	PRS Channel 8 selected
9	PRSCH9	PRS Channel 9 selected
10	PRSCH10	PRS Channel 10 selected
11	PRSCH11	PRS Channel 11 selected
12	PRSCH12	PRS Channel 12 selected
13	PRSCH13	PRS Channel 13 selected
14	PRSCH14	PRS Channel 14 selected
15	PRSCH15	PRS Channel 15 selected

5:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

15.5.11 PRS_PEEK - PRS Channel Values

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																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access																	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name																	CH15VAL	CH14VAL	CH13VAL	CH12VAL	CH11VAL	CH10VAL	CH9VAL	CH8VAL	CH7VAL	CH6VAL	CH5VAL	CH4VAL	CH3VAL	CH2VAL	CH1VAL	CH0VAL

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	CH15VAL	0	R	**Channel 15 Current Value**
	See bit 0.			
14	CH14VAL	0	R	**Channel 14 Current Value**
	See bit 0.			
13	CH13VAL	0	R	**Channel 13 Current Value**
	See bit 0.			
12	CH12VAL	0	R	**Channel 12 Current Value**
	See bit 0.			
11	CH11VAL	0	R	**Channel 11 Current Value**
	See bit 0.			
10	CH10VAL	0	R	**Channel 10 Current Value**
	See bit 0.			
9	CH9VAL	0	R	**Channel 9 Current Value**
	See bit 0.			
8	CH8VAL	0	R	**Channel 8 Current Value**
	See bit 0.			
7	CH7VAL	0	R	**Channel 7 Current Value**
	See bit 0.			
6	CH6VAL	0	R	**Channel 6 Current Value**
	See bit 0.			
5	CH5VAL	0	R	**Channel 5 Current Value**
	See bit 0.			
4	CH4VAL	0	R	**Channel 4 Current Value**
	See bit 0.			
3	CH3VAL	0	R	**Channel 3 Current Value**
	See bit 0.			

Bit	Name	Reset	Access	Description
2	CH2VAL	0	R	**Channel 2 Current Value**
	See bit 0.			
1	CH1VAL	0	R	**Channel 1 Current Value**
	See bit 0.			
0	CH0VAL	0	R	**Channel 0 Current Value**
	When ASYNC = 0, sample the current output value of channel 0. Any enabled edge detection will not be visible. This value may be one or two clock delayed. When ASYNC = 1, no value is returned			

15.5.12 PRS_CHx_CTRL - Channel 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		0		0	0	0	0				0x0							0x00												0x0		
Access		RW		RW	RW	RW	RW				RW							RW												RW		
Name		ASYNC		ANDNEXT	ORPREV	INV	STRETCH				EDSEL							SOURCESEL												SIGSEL		

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	ASYNC	0	RW	**Asynchronous Reflex**
	Set to enable asynchronous mode of this reflex signal			
29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28	ANDNEXT	0	RW	**And Next**
	If set, channel output is AND'ed with the next channel output			
27	ORPREV	0	RW	**Or Previous**
	If set, channel output is OR'ed with the previous channel output			
26	INV	0	RW	**Invert Channel**
	If set, channel output is inverted			
25	STRETCH	0	RW	**Stretch Channel Output**
	If set, stretches channel output to ensure that the target clock domain sees it.			
24:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	EDSEL	0x0	RW	**Edge Detect Select**
	Select edge detection.			

Value	Mode	Description
0	OFF	Signal is left as it is
1	POSEDGE	A one HFCLK cycle pulse is generated for every positive edge of the incoming signal
2	NEGEDGE	A one HFCLK clock cycle pulse is generated for every negative edge of the incoming signal
3	BOTHEDGES	A one HFCLK clock cycle pulse is generated for every edge of the incoming signal

19:15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
14:8	SOURCESEL	0x00	RW	**Source Select**

Select input source to PRS channel.

Value	Mode	Description
0b0000000	NONE	No source selected
0b0000001	PRSL	Peripheral Reflex System
0b0000010	PRS	Peripheral Reflex System
0b0000100	ACMP0	Analog Comparator 0
0b0000101	ACMP1	Analog Comparator 1
0b0000110	ADC0	Analog to Digital Converter 0
0b0000111	RTC	Real-Time Counter
0b0001000	RTCC	Real-Time Counter and Calendar
0b0001001	GPIOL	General purpose Input/Output
0b0001010	GPIOH	General purpose Input/Output
0b0001011	LETIMER0	Low Energy Timer 0
0b0001100	LETIMER1	Low Energy Timer 1
0b0001101	PCNT0	Pulse Counter 0
0b0001110	PCNT1	Pulse Counter 1
0b0001111	PCNT2	Pulse Counter 2
0b0010000	CRYOTIMER	CRYOTIMER
0b0010001	CMU	Clock Management Unit
0b0010111	VDAC0	Digital to Analog Converter 0
0b0011000	LESENSEL	Low Energy Sensor Interface
0b0011001	LESENSEH	Low Energy Sensor Interface
0b0011010	LESENSED	Low Energy Sensor Interface
0b0011011	LESENSE	Low Energy Sensor Interface
0b0011100	ACMP2	Analog Comparator 2
0b0011101	ADC1	Analog to Digital Converter 0
0b0110000	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter 0
0b0110001	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter 1
0b0110010	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter 2
0b0110011	USART3	Universal Synchronous/Asynchronous Receiver/Transmitter 3
0b0110100	USART4	Universal Synchronous/Asynchronous Receiver/Transmitter 4
0b0110110	UART0	Universal Asynchronous Receiver/Transmitter 0
0b0110111	UART1	Universal Asynchronous Receiver/Transmitter 1
0b0111100	TIMER0	Timer 0
0b0111101	TIMER1	Timer 1
0b0111110	TIMER2	Timer 2

Bit	Name	Reset	Access	Description
	0b1000000	USB		Universal Serial Bus Interface
	0b1000011	CM4		
	0b1010000	TIMER3		Timer 3
	0b1010010	WTIMER0		Wide Timer 0
	0b1010011	WTIMER1		Wide Timer 0
	0b1111001	PDM		PDM Interface
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 PRS channel. Selected signal depends on SOURCESEL as indicated.

Value	Mode	Description
SOURCESEL =	*0b000000*	*(NONE)*
0bxxx	OFF	Channel input selection is turned off
SOURCESEL =	*0b0000001*	*(PRSL)*
0b000	PRSCH0	PRS channel 0 PRSCH0 (Asynchronous)
0b001	PRSCH1	PRS channel 1 PRSCH1 (Asynchronous)
0b010	PRSCH2	PRS channel 2 PRSCH2 (Asynchronous)
0b011	PRSCH3	PRS channel 3 PRSCH3 (Asynchronous)
0b100	PRSCH4	PRS channel 4 PRSCH4 (Asynchronous)
0b101	PRSCH5	PRS channel 5 PRSCH5 (Asynchronous)
0b110	PRSCH6	PRS channel 6 PRSCH6 (Asynchronous)
0b111	PRSCH7	PRS channel 7 PRSCH7 (Asynchronous)
SOURCESEL =	*0b0000010*	*(PRS)*
0b000	PRSCH8	PRS channel 8 PRSCH8 (Asynchronous)
0b001	PRSCH9	PRS channel 9 PRSCH9 (Asynchronous)
0b010	PRSCH10	PRS channel 10 PRSCH10 (Asynchronous)
0b011	PRSCH11	PRS channel 11 PRSCH11 (Asynchronous)
0b100	PRSCH12	PRS channel 12 PRSCH12 (Asynchronous)
0b101	PRSCH13	PRS channel 13 PRSCH13 (Asynchronous)
0b110	PRSCH14	PRS channel 14 PRSCH14 (Asynchronous)
0b111	PRSCH15	PRS channel 15 PRSCH15 (Asynchronous)
SOURCESEL =	*0b0000100*	*(ACMP0)*
0b000	ACMP0OUT	Analog comparator output ACMP0OUT (Asynchronous)
SOURCESEL =	*0b0000101*	*(ACMP1)*
0b000	ACMP1OUT	Analog comparator output ACMP1OUT (Asynchronous)
SOURCESEL =	*0b0000110*	*(ADC0)*
0b000	ADC0SINGLE	ADC single conversion done ADC0SINGLE (Asynchronous)

Bit	Name	Reset	Access	Description
	0b001	ADC0SCAN		ADC scan conversion done ADC0SCAN (Asynchronous)
	SOURCESEL =	*0b0000111*		*(RTC)*
	0b000	RTCOF		RTC Overflow RTCOF (Asynchronous)
	0b001	RTCCOMP0		RTC Compare 0 RTCCOMP0 (Asynchronous)
	0b010	RTCCOMP1		RTC Compare 1 RTCCOMP1 (Asynchronous)
	0b011	RTCCOMP2		RTC Compare 2 RTCCOMP2 (Asynchronous)
	0b100	RTCCOMP3		RTC Compare 3 RTCCOMP3 (Asynchronous)
	0b101	RTCCOMP4		RTC Compare 4 RTCCOMP4 (Asynchronous)
	0b110	RTCCOMP5		RTC Compare 5 RTCCOMP5 (Asynchronous)
	SOURCESEL =	*0b0001000*		*(RTCC)*
	0b001	RTCCCCV0		RTCC Compare 0 RTCCCCV0 (Asynchronous)
	0b010	RTCCCCV1		RTCC Compare 1 RTCCCCV1 (Asynchronous)
	0b011	RTCCCCV2		RTCC Compare 2 RTCCCCV2 (Asynchronous)
	SOURCESEL =	*0b0001001*		*(GPIOL)*
	0b000	GPIOPIN0		GPIO pin 0 GPIOPIN0 (Asynchronous)
	0b001	GPIOPIN1		GPIO pin 1 GPIOPIN1 (Asynchronous)
	0b010	GPIOPIN2		GPIO pin 2 GPIOPIN2 (Asynchronous)
	0b011	GPIOPIN3		GPIO pin 3 GPIOPIN3 (Asynchronous)
	0b100	GPIOPIN4		GPIO pin 4 GPIOPIN4 (Asynchronous)
	0b101	GPIOPIN5		GPIO pin 5 GPIOPIN5 (Asynchronous)
	0b110	GPIOPIN6		GPIO pin 6 GPIOPIN6 (Asynchronous)
	0b111	GPIOPIN7		GPIO pin 7 GPIOPIN7 (Asynchronous)
	SOURCESEL =	*0b0001010*		*(GPIOH)*
	0b000	GPIOPIN8		GPIO pin 8 GPIOPIN8 (Asynchronous)
	0b001	GPIOPIN9		GPIO pin 9 GPIOPIN9 (Asynchronous)
	0b010	GPIOPIN10		GPIO pin 10 GPIOPIN10 (Asynchronous)
	0b011	GPIOPIN11		GPIO pin 11 GPIOPIN11 (Asynchronous)
	0b100	GPIOPIN12		GPIO pin 12 GPIOPIN12 (Asynchronous)
	0b101	GPIOPIN13		GPIO pin 13 GPIOPIN13 (Asynchronous)
	0b110	GPIOPIN14		GPIO pin 14 GPIOPIN14 (Asynchronous)
	0b111	GPIOPIN15		GPIO pin 15 GPIOPIN15 (Asynchronous)
	SOURCESEL =	*0b0001011*		*(LETIMER0)*
	0b000	LETIMER0CH0		LETIMER CH0 Out LETIMER0CH0 (Asynchronous)
	0b001	LETIMER0CH1		LETIMER CH1 Out LETIMER0CH1 (Asynchronous)
	SOURCESEL =	*0b0001100*		*(LETIMER1)*
	0b000	LETIMER1CH0		LETIMER CH0 Out LETIMER1CH0 (Asynchronous)
	0b001	LETIMER1CH1		LETIMER CH1 Out LETIMER1CH1 (Asynchronous)

Bit	Name	Reset	Access	Description
	SOURCESEL =	*0b0001101*		*(PCNT0)*
	0b000	PCNT0TCC		Triggered compare match PCNT0TCC (Asynchronous)
	0b001	PCNT0UFOF		Counter overflow or underflow PCNT0UFOF (Asynchronous)
	0b010	PCNT0DIR		Counter direction PCNT0DIR (Asynchronous)
	SOURCESEL =	*0b0001110*		*(PCNT1)*
	0b000	PCNT1TCC		Triggered compare match PCNT1TCC (Asynchronous)
	0b001	PCNT1UFOF		Counter overflow or underflow PCNT1UFOF (Asynchronous)
	0b010	PCNT1DIR		Counter direction PCNT1DIR (Asynchronous)
	SOURCESEL =	*0b0001111*		*(PCNT2)*
	0b000	PCNT2TCC		Triggered compare match PCNT2TCC (Asynchronous)
	0b001	PCNT2UFOF		Counter overflow or underflow PCNT2UFOF (Asynchronous)
	0b010	PCNT2DIR		Counter direction PCNT2DIR (Asynchronous)
	SOURCESEL =	*0b0010000*		*(CRYOTIMER)*
	0b000	CRYOTIMERPERIOD		CRYOTIMER Output CRYOTIMERPERIOD (Asynchronous)
	SOURCESEL =	*0b0010001*		*(CMU)*
	0b000	CMUCLKOUT0		Clock Output 0 CMUCLKOUT0 (Asynchronous)
	0b001	CMUCLKOUT1		Clock Output 1 CMUCLKOUT1 (Asynchronous)
	0b111	CMUCLKOUT2		Clock Output 2 CMUCLKOUT2 (Asynchronous)
	SOURCESEL =	*0b0010111*		*(VDAC0)*
	0b000	VDAC0CH0		DAC ch0 conversion done VDAC0CH0
	0b001	VDAC0CH1		DAC ch1 conversion done VDAC0CH1
	0b010	VDAC0OPA0		OPA0 warmed up. output is valid. VDAC0OPA0 (Asynchronous)
	0b011	VDAC0OPA1		OPA1 warmed up. output is valid. VDAC0OPA1 (Asynchronous)
	0b100	VDAC0OPA2		OPA2 warmed up. output is valid. VDAC0OPA2 (Asynchronous)
	0b101	VDAC0OPA3		OPA3 warmed up. output is valid. VDAC0OPA3 (Asynchronous)
	SOURCESEL =	*0b0011000*		*(LESENSEL)*
	0b000	LESENSESCANRES0		LESENSE SCANRES register, bit 0 LESENSESCANRES0 (Asynchronous)
	0b001	LESENSESCANRES1		LESENSE SCANRES register, bit 1 LESENSESCANRES1 (Asynchronous)
	0b010	LESENSESCANRES2		LESENSE SCANRES register, bit 2 LESENSESCANRES2 (Asynchronous)
	0b011	LESENSESCANRES3		LESENSE SCANRES register, bit 3 LESENSESCANRES3 (Asynchronous)
	0b100	LESENSESCANRES4		LESENSE SCANRES register, bit 4 LESENSESCANRES4 (Asynchronous)
	0b101	LESENSESCANRES5		LESENSE SCANRES register, bit 5 LESENSESCANRES5 (Asynchronous)
	0b110	LESENSESCANRES6		LESENSE SCANRES register, bit 6 LESENSESCANRES6 (Asynchronous)

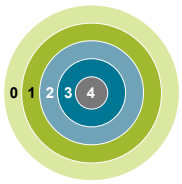
Bit	Name	Reset	Access	Description
	0b111	LESENSESCANRES7		LESENSE SCANRES register, bit 7 LESENSESCANRES7 (Asynchronous)
	SOURCESEL =	*0b0011001*		*(LESENSEH)*
	0b000	LESENSESCANRES8		LESENSE SCANRES register, bit 8 LESENSESCANRES8 (Asynchronous)
	0b001	LESENSESCANRES9		LESENSE SCANRES register, bit 9 LESENSESCANRES9 (Asynchronous)
	0b010	LESENSESCANRES10		LESENSE SCANRES register, bit 10 LESENSESCANRES10 (Asynchronous)
	0b011	LESENSESCANRES11		LESENSE SCANRES register, bit 11 LESENSESCANRES11 (Asynchronous)
	0b100	LESENSESCANRES12		LESENSE SCANRES register, bit 12 LESENSESCANRES12 (Asynchronous)
	0b101	LESENSESCANRES13		LESENSE SCANRES register, bit 13 LESENSESCANRES13 (Asynchronous)
	0b110	LESENSESCANRES14		LESENSE SCANRES register, bit 14 LESENSESCANRES14 (Asynchronous)
	0b111	LESENSESCANRES15		LESENSE SCANRES register, bit 15 LESENSESCANRES15 (Asynchronous)
	SOURCESEL =	*0b0011010*		*(LESENSED)*
	0b000	LESENSEDEC0		LESENSE Decoder PRS out 0 LESENSEDEC0 (Asynchronous)
	0b001	LESENSEDEC1		LESENSE Decoder PRS out 1 LESENSEDEC1 (Asynchronous)
	0b010	LESENSEDEC2		LESENSE Decoder PRS out 2 LESENSEDEC2 (Asynchronous)
	0b011	LESENSEDECCMP		LESENSE Decoder PRS compare value match channel LESENSEDECCMP (Asynchronous)
	SOURCESEL =	*0b0011011*		*(LESENSE)*
	0b000	LESENSEMEASACT		LESENSE Measurement active LESENSEMEASACT (Asynchronous)
	SOURCESEL =	*0b0011100*		*(ACMP2)*
	0b000	ACMP2OUT		Analog comparator output ACMP2OUT (Asynchronous)
	SOURCESEL =	*0b0011101*		*(ADC1)*
	0b000	ADC1SINGLE		ADC single conversion done ADC1SINGLE (Asynchronous)
	0b001	ADC1SCAN		ADC scan conversion done ADC1SCAN (Asynchronous)
	SOURCESEL =	*0b0110000*		*(USART0)*
	0b000	USART0IRTX		USART0IRTX
	0b001	USART0TXC		USART0TXC
	0b010	USART0RXDATAV		USART0RXDATAV
	0b011	USART0RTS		USART0RTS
	0b101	USART0TX		USART0TX
	0b110	USART0CS		USART0CS
	SOURCESEL =	*0b0110001*		*(USART1)*
	0b001	USART1TXC		USART1TXC

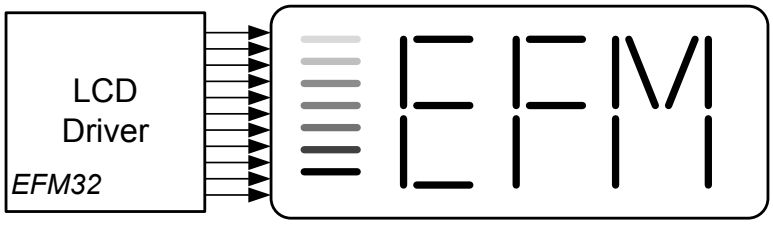
Bit	Name	Reset	Access	Description
	0b010	USART1RXDATAV		USART1RXDATAV
	0b011	USART1RTS		USART1RTS
	0b101	USART1TX		USART1TX
	0b110	USART1CS		USART1CS
	SOURCESEL =	*0b0110010*		*(USART2)*
	0b000	USART2IRTX		USART 2 IRDA out USART2IRTX (Asynchronous)
	0b001	USART2TXC		USART2TXC
	0b010	USART2RXDATAV		USART2RXDATAV
	0b011	USART2RTS		USART2RTS
	0b101	USART2TX		USART2TX
	0b110	USART2CS		USART2CS
	SOURCESEL =	*0b0110011*		*(USART3)*
	0b001	USART3TXC		USART3TXC
	0b010	USART3RXDATAV		USART3RXDATAV
	0b011	USART3RTS		USART3RTS
	0b101	USART3TX		USART3TX
	0b110	USART3CS		USART3CS
	SOURCESEL =	*0b0110100*		*(USART4)*
	0b001	USART4TXC		USART4TXC
	0b010	USART4RXDATAV		USART4RXDATAV
	0b011	USART4RTS		USART4RTS
	0b101	USART4TX		USART4TX
	0b110	USART4CS		USART4CS
	SOURCESEL =	*0b0110110*		*(UART0)*
	0b001	UART0TXC		UART0TXC
	0b010	UART0RXDATAV		UART0RXDATAV
	0b011	UART0RTS		UART0RTS
	0b101	UART0TX		UART0TX
	0b110	UART0CS		UART0CS
	SOURCESEL =	*0b0110111*		*(UART1)*
	0b001	UART1TXC		UART1TXC
	0b010	UART1RXDATAV		UART1RXDATAV
	0b011	UART1RTS		UART1RTS
	0b101	UART1TX		UART1TX
	0b110	UART1CS		UART1CS
	SOURCESEL =	*0b0111100*		*(TIMER0)*
	0b000	TIMER0UF		TIMER0UF

Bit	Name	Reset	Access	Description
	0b001	TIMER0OF		TIMER0OF
	0b010	TIMER0CC0		TIMER0CC0
	0b011	TIMER0CC1		TIMER0CC1
	0b100	TIMER0CC2		TIMER0CC2
	SOURCESEL =	*0b0111101*		*(TIMER1)*
	0b000	TIMER1UF		TIMER1UF
	0b001	TIMER1OF		TIMER1OF
	0b010	TIMER1CC0		TIMER1CC0
	0b011	TIMER1CC1		TIMER1CC1
	0b100	TIMER1CC2		TIMER1CC2
	0b101	TIMER1CC3		TIMER1CC3
	SOURCESEL =	*0b0111110*		*(TIMER2)*
	0b000	TIMER2UF		TIMER2UF
	0b001	TIMER2OF		TIMER2OF
	0b010	TIMER2CC0		TIMER2CC0
	0b011	TIMER2CC1		TIMER2CC1
	0b100	TIMER2CC2		TIMER2CC2
	SOURCESEL =	*0b1000000*		*(USB)*
	0b000	USBSOF		USB Start of Frame USBSOF (Asynchronous)
	0b001	USBSOFSR		USB Start of Frame Sent/Received USBSOFSR (Asynchronous)
	SOURCESEL =	*0b1000011*		*(CM4)*
	0b000	CM4TXEV		CM4TXEV
	0b001	CM4ICACHEPCHIT-SOF		CM4ICACHEPCHITSOF
	0b010	CM4ICACHEPCMISSE-SOF		CM4ICACHEPCMISSESOF
	SOURCESEL =	*0b1010000*		*(TIMER3)*
	0b000	TIMER3UF		TIMER3UF
	0b001	TIMER3OF		TIMER3OF
	0b010	TIMER3CC0		TIMER3CC0
	0b011	TIMER3CC1		TIMER3CC1
	0b100	TIMER3CC2		TIMER3CC2
	SOURCESEL =	*0b1010010*		*(WTIMER0)*
	0b000	WTIMER0UF		WTIMER0UF
	0b001	WTIMER0OF		WTIMER0OF
	0b010	WTIMER0CC0		WTIMER0CC0
	0b011	WTIMER0CC1		WTIMER0CC1
	0b100	WTIMER0CC2		WTIMER0CC2

Bit	Name	Reset	Access	Description
	SOURCESEL =	*0b1010011*		*(WTIMER1)*
	0b000	WTIMER1UF		WTIMER1UF
	0b001	WTIMER1OF		WTIMER1OF
	0b010	WTIMER1CC0		WTIMER1CC0
	0b011	WTIMER1CC1		WTIMER1CC1
	0b100	WTIMER1CC2		WTIMER1CC2
	0b101	WTIMER1CC3		WTIMER1CC3
	SOURCESEL =	*0b1111001*		*(PDM)*
	0b000	PDMDSRPULSE		PDM DSR pulse PDMDSRPULSE (Asynchronous)

16. LCD - Liquid Crystal Display Driver





Quick Facts

What?

The LCD driver can drive LCD displays of up to 8x36 segments. The animation feature makes it possible to have active animations without CPU intervention.

Why?

Segmented LCD displays are a common way to display information. The extreme low-power LCD driver enables a lot of applications to utilize an LCD display even in energy critical systems.

How?

The low frequency clock signal, low-power waveform, animation and blink capabilities enable the LCD driver to run autonomously in EM2 DeepSleep for long periods. Adding the flexible frame rate setting, contrast control, and different multiplexing modes make the EFM32 Giant Gecko 12 the optimal choice for battery-driven systems with LCD panels.

16.1 Introduction

The LCD driver is capable of driving a segmented LCD display combination of: 1x40, 2x40, 3x40, 4x40, 6x38 or 8x36 segments. A charge pump enables it to provide the LCD display with higher voltage than the supply voltage for the device. In addition, an animation feature can run custom animations on the LCD display without any CPU intervention. The LCD driver can also remain active in Energy Mode 2 and provides a Frame Counter interrupt that can wake-up the device on a regular basis for updating data.

16.2 Features

- Up to 4x40 or 8x36 segments.
- Configurable multiplexing (1, 2, 3, 4, 6, 8)
- LCD supports the following COM/SEG combinations
  - 1x40, 2x40, 3x40, 4x40, 6x38 or 8x36
- Configurable bias/voltage levels settings
- Configurable clock source prescaler
- Configurable Frame rate
- Segment lines can be enabled or disabled individually
- Blink capabilities
- Integrated animation functionality
  - Available on SEG0-SEG7 or SEG8-SEG15
- Charge redistribution feature reduces LCD module current consumption by up to 40%
- Charge pump
- Programmable contrast
- Frame Counter
- LCD frame interrupt
- Direct segment control

16.3 Functional Description

An overview of the LCD module is shown in Figure 16.1 LCD Block Diagram on page 571. The module provides the necessary waveforms for turning each segment of an LCD display on or off.

The waveforms are multiplexed between eight (1-8) different common lines and segment lines to support up to 288 different LCD segments. The common lines and segment lines can be enabled or disabled individually to prevent the LCD driver from occupying more I/O resources than required.

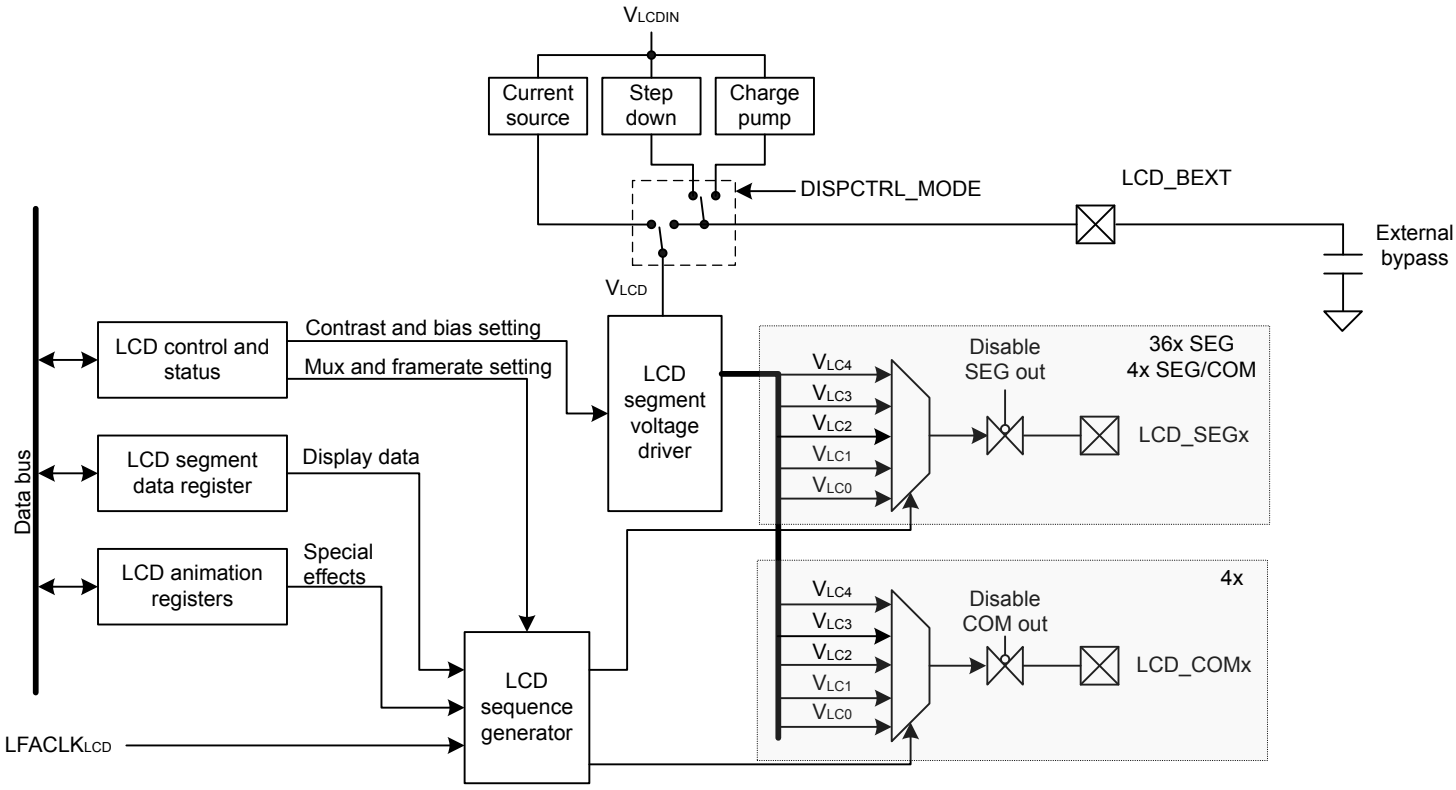


Figure 16.1. LCD Block Diagram

16.3.1 Power Supply

The LCD block power (V_{LCDIN}) is derived from the VDDX_ANA supply rail. VDDX_ANA can be selected from the AVDD or DVDD supply pins using the EMU_PWRCTRL_ANASW bit field. The LCD block generates its own stable supply (V_{LCD}), which is used to derive all output voltages.

16.3.2 LCD Driver Enable

Setting the EN bit in LCD_CTRL enables the LCD driver. The MUX bit-field in LCD_DISPCTRL determines which COM lines are driven by the LCD driver. By default, LCD_COM0 is driven whenever the LCD driver is enabled. The LCD_SEGEN and SEGEN2 registers determine which segment lines are enabled. Individual segment lines can be enabled or disabled.

Each pin being used by the LCD block should be set to the DISABLED state in the GPIO block. Any other GPIO setting will prevent the LCD controller from accessing the pin. See the device data sheet for a mapping of LCD signals to GPIO pins.

16.3.3 LCD Frame Rate and Power Reduction

LCD Frame rates are usually set between 30 to 100 frames per second (FPS). The $LFCLK_{LCD}$ can be prescaled in the CMU. FRDIV in LCD_FRAMERATE is used to further divide the LCD clock rate and provides the final frame rate. The power consumption of the LCD panel itself can be lowered through the use of charge redistribution. When charge redistribution is in use, all segments are briefly shorted together, allowing low segments to be partially charged by the energy in high segments instead of using additional energy from the power supply. A Static MUX selection will have two phase periods per frame, while an Octaplex MUX selection will have 16 phase periods per frame.

CHGRDST in LCD_DISPCTRL is used to select the number of prescaled $LFCLK_{LCD}$ cycles used for charge redistribution. Refer to Figure 16.2 Charge Redistribution Cycle Percentage on page 572 to calculate the charge redistribution cycle percentage (CHGRDST PERCENT).

CHGRDST PERCENT = CHGRDST / FR Divider

Figure 16.2. Charge Redistribution Cycle Percentage

The charge redistribution cycle percentage should be 5% or less to prevent reduction in LCD pad RMS voltage. If charge redistribution is not used, a larger CMU prescaling value is recommended to minimize power consumed by the LCD block itself. Note that disabling charge redistribution will always result in higher system power consumption. Charge redistribution is on by default, but it can be disabled by setting CHGRDST to disable in LCD_DISPCTRL. Refer to table Table 16.1 LCD Frame Rate on page 572 for examples of the percentage of time that charge redistribution is on in various scenarios using a 32 kHz clock.

Table 16.1. LCD Frame Rate

MUX	CHGRDST	CMU Prescaler	FR Divider	FPS	CHGRDST PERCENT
Static	4	DIV1	512	32.0	0.8%
Static	4	DIV1	163	100.5	2.5%
Static	1	DIV16	32	32.0	3.1%
Static	1	DIV8	20	102.4	5.0%
Static	0	DIV128	4	32.0	0.0%
Static	0	DIV32	5	102.4	0.0
Quadruplex	4	DIV1	136	30.1	2.9%
Quadruplex	2	DIV1	40	102.4	5.0%
Quadruplex	1	DIV4	34	30.1	2.9%
Quadruplex	1	DIV2	20	102.4	5.0%
Quadruplex	0	DIV16	8	32.0	0.0%
Quadruplex	0	DIV4	10	102.4	0.0%
Octaplex	2	DIV1	68	30.1	2.9%
Octaplex	1	DIV1	20	102.4	5.0%
Octaplex	1	DIV2	34	30.1	2.9%
Octaplex	0	DIV64	1	32.0	0.0%

16.3.4 Multiplexing, Bias, and Wave Settings

The LCD driver supports different multiplexing and bias settings, and these can be set individually in the MUX and BIAS bits in LCD_DISPCTRL respectively, see Table 16.2 LCD Mux Settings on page 573 and Table 16.3 LCD BIAS Settings on page 573.

Note: If the MUX and BIAS settings in LCD_DISPCTRL are changed while the LCD driver is enabled, the output waveform is unpredictable and may lead to a DC-component for one LCD frame.

The MUX setting determines the number of LCD COM lines that are enabled. When using octaplex or sextaplex multiplexing, the additional COM lines used (COM4-COM7) are actually located on the (SEG20-SEG23) lines. When static multiplexing is selected, LCD output is enabled on LCD_COM0, when duplex multiplexing is used, LCD_COM0-LCD_COM1 are used, when triplex multiplexing is selected, LCD_COM0-LCD_COM2 are used, when quadruplex multiplexing is selected, LCD_COM0-LCD_COM3 are used, when sextaplex multiplexing is selected, LCD_COM0-LCD_COM3 and SEG20-SEG21 act as common pins, reducing the number of available segment pins to 38 . Finally when octaplex multiplexing is selected, LCD_COM0-LCD_COM3 and SEG20-SEG23 act as common pins, reducing the number of available segment pins to 36 .

See 16.3.15 Waveform Examples for waveforms for the different bias and multiplexing settings.

The waveforms generated by the LCD controller can be generated in two different versions, regular and low-power. The low power mode waveforms have a lower switching frequency than the regular waveforms, and thus consume less power. The WAVE bit in LCD_DISPCTRL decides which waveforms to generate. An example of a low-power waveform is shown in Figure 16.3 LCD Low-power Waveform for LCD_COM0 in Quadruples Multiplex Mode, 1/3 Bias on page 574, and an example of a regular waveform is shown in Figure 16.4 LCD Normal Waveform for LCD_COM0 in Quadruples Multiplex Mode, 1/3 Bias on page 574. For COM waveforms, a green dotted lines indicates where the SEG waveform would be for an 'on' level while the red dotted lines indicate where the SEG waveform would be for an 'off' level.

Table 16.2. LCD Mux Settings

MUX	Mode	Multiplexing
000	Static	Static (segments can be multiplexed with LCD_COM[0])
001	Duplex	Duplex (segments can be multiplexed with LCD_COM[1:0])
010	Triplex	Triplex (segments can be multiplexed with LCD_COM[2:0])
011	Quadruplex	Quadruplex (segments can be multiplexed with LCD_COM[3:0])
101	Sextaplex	Sextaplex (segments can be multiplexed with LCD_COM[3:0] and SEG[21:20])
111	Octaplex	Octaplex (segments can be multiplexed with LCD_COM[3:0]) and SEG[23:20]

Table 16.3. LCD BIAS Settings

BIAS	Mode	Bias setting
00	Static	Static (2 levels)
01	Half Bias	1/2 Bias (3 levels)
10	Third Bias	1/3 Bias (4 levels)
11	Fourth Bias	1/4 Bias (5 levels)

Table 16.4. LCD Wave Settings

WAVE	Mode	Wave mode
0	LowPower	Low power optimized waveform output
1	Normal	Regular waveform output

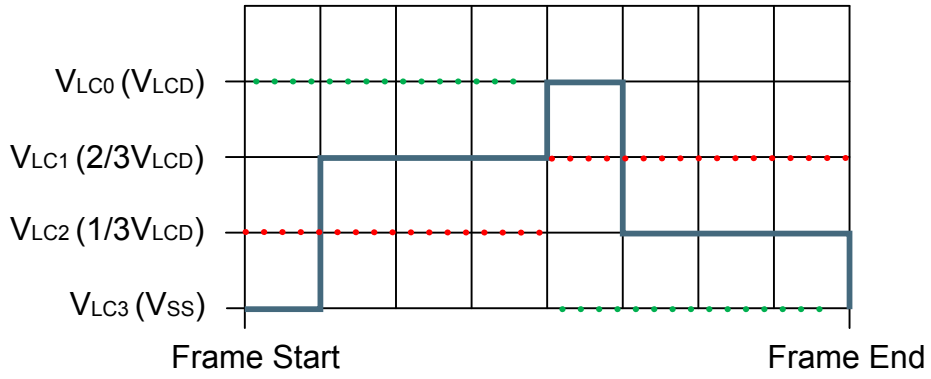


Figure 16.3. LCD Low-power Waveform for LCD_COM0 in Quadruples Multiplex Mode, 1/3 Bias

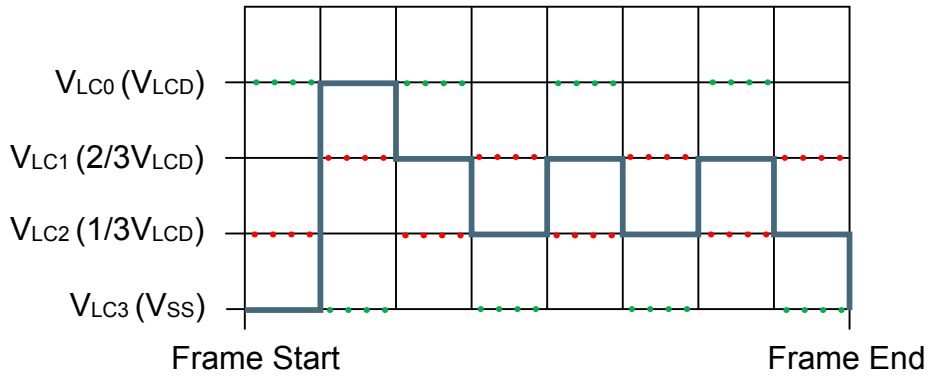


Figure 16.4. LCD Normal Waveform for LCD_COM0 in Quadruples Multiplex Mode, 1/3 Bias

16.3.5 LCD Contrast

To compensate for variations in LCD panels, the LCD driver has a programmable contrast that adjusts V_{LCD}. The contrast is set by CONTRAST in LCD_DISPCTRL.

Table 16.5. LCD Contrast

Mode	Equation for V_{LCD}
current source	(CONTRAST[4:0] + 25) * 68.6mV
step down and charge pump	(CONTRAST[5:0] + 24) * 43.3mV

16.3.6 Voltage Levels and Mode Selection

By default, V_{LCD} is powered from the AVDD pin. If the ANASW bitfield in EMU->PWRCTRL is set, V_{LCD} will be powered from the DVDD pin instead. However, if using the DC-DC Converter the DVDD pin will typically be at 1.8 V, and thus not suitable for driving an LCD display directly.

The number of LCD bias levels is controlled by BIAS field in LCD_DISPCTRL. When BIAS is set to ONEFOURTH, voltages of V_{LCD}, ¾ V_{LCD}, ½ V_{LCD}, ¼ V_{LCD}, and VSS are generated. For a BIAS setting of ONETHIRD, the generated voltages are V_{LCD}, 2/3 V_{LCD}, 1/3 V_{LCD}, and VSS. For a BIAS setting of ONEHALF, the generated voltages are V_{LCD}, 1/2 V_{LCD}, and VSS. For a BIAS setting of STATIC, the voltage are only V_{LCD} and VSS.

Three modes are available for setting the V_{LCD} level: current source mode, step down mode, or charge pump mode. For the current source mode no external capacitor is used. An internal current source is adjusted using CONTRAST[4:0] in LCD_DISPCTRL to set the V_{LCD} voltage level.

For the step down mode an external capacitor is regulated using an LCD comparator to maintain a V_{LCD} voltage that is not greater than the supply voltage.

For the charge pump mode, a voltage of up to twice the supply voltage is generated internally and maintained on an external capacitor, which maintains the V_{LCD} voltage.

In both the step down and charge pump modes, the LCD_BEXT signal is used. The LCD_BEXT pin should be connected through a capacitor to VSS. For most applications, a 1 µF capacitor is sufficient to prevent any visible artifacts from supply ripple. However, larger capacitors may be used to reduce the supply ripple if needed. The recommended value is approximately 1000 times the total LCD segment capacitance.

Note: All LCD pins should have the OVT protection enabled using GPIO_Px_OVTDIS registers before the LCD is enabled.The segment/com enables should not be changed while the LCD is enabled.

16.3.7 Frame Rate

It is important to choose the correct frame rate for the LCD display. Normally, the frame rate should be between 30 and 100 Hz. A frame rate below 30 Hz may lead to flickering, while a frame rate above 100 Hz may lead to ghostring and unnecessarily high power consumption.

16.3.7.1 Clock Selection and Prescaler

The LFACLK is prescaled to $LFACLK_{LCDpre}$ in the CMU. In addition to selecting the correct prescaling, the clock source can be selected in the CMU. To use this module, the LE interface clock must be enabled in CMU_HFBUSCLKEN0, in addition to the module clock.

16.3.7.2 Frame Rate Division Register

The frame rate is set with FRDIV in LCD_FRAMERATE. FRDIV sets the frame rate phase frequency and the number of phases per frame is determined by the multiplex setting. Each COM line requires two phases. This setting should not be changed while the LCD driver is running. The equation for calculating the resulting frame rate is given from Figure 16.5 LCD Frame rate Calculation on page 576

$$LFACLK_{LCD} = LFACLK_{LCDpre}/(1 + FRDIV)$$

Figure 16.5. LCD Frame rate Calculation

16.3.8 Data Update

The LCD driver logic that controls the output waveforms is clocked on $LFACLK_{LCDpre}$. The LCD data and Control Registers are clocked on the HFCORECLK. Segment data should not be changed in the middle of a frame. The LCD driver has functionality to synchronize data transfer to the LCD frames. The synchronization logic is applied to all data that need to be updated at the beginning of the LCD frames:

- LCD_SEGDn
- LCD_AREGA
- LCD_AREGB
- LCD_BACTRL

The different methods to update data are controlled by the UDCTRL bits in LCD_CTRL.

Table 16.6. LCD Update Data Control (UDCTRL) Bits

UDCTRL	Mode	Description
00	REGULAR	The data transfer is controlled by SW and data synchronization is initiated by writing data to the buffers. Data is transferred as soon as possible, possibly creating a frame with a DC component on the LCD.
01	FCEVENT	The data transfer is done at the next event triggered by the Frame Counter (FC). See 16.3.10 Frame Counter (FC) for details on how to configure the Frame Counter. Optionally, the Frame Counter can also generate an interrupt at every event.
10	FRAMESTART	The data transfer is done at frame-start.

16.3.9 Direct Segment Control (DSC)

It is possible to gain direct control over the bias levels for each SEG/COM line by setting DSC in LCD_CTRL, overwriting the BIAS settings in LCD_DISPCTRL. The SEG lines bias levels can be set in SEGD0-SEGD3, while the COM line bias levels can be set in SEGD4. To represent the different bias levels, 4-bits per SEG lines are needed. For example, SEG0's bias levels can be set using SEGD0[3:0], and SEG1 can be controlled through SEGD1[3:0] etc. Bias level encoding is shown in Table 16.7 DSC BIAS Encoding on page 577, and segment/common locations are shown in Table 16.8 DSC Segment and Common Mapping on page 577.

Table 16.7. DSC BIAS Encoding

SEGD	Bias setting
0000	tristate
0001	VSS
0010	1/3 or 1/4 V_{LCD}
0011	1/2 V_{LCD}
0100	2/3 or 3/4 V_{LCD}
0101	V_{LCD}

Table 16.8. DSC Segment and Common Mapping

Register	H[7:4]	H[3:0]	L[31:28]	L[27:24]	L[23:20]	L[19:16]	L[15:12]	L[11:8]	L[7:4]	L[3:0]
SEGD0	seg36	seg32	seg28	seg24	seg20	seg16	seg12	seg8	seg4	seg0
SEGD1	seg37	seg33	seg29	seg25	seg21	seg17	seg13	seg9	seg5	seg1
SEGD2	seg38	seg34	seg30	seg26	seg22	seg18	seg14	seg10	seg6	seg2
SEGD3	seg39	seg35	seg31	seg27	seg23	seg19	seg15	seg11	seg7	seg3
SEGD4							com3	com2	com1	com0

16.3.10 Frame Counter (FC)

The Frame Counter is synchronized to the LCD frame start and will generate an event after a programmable number of frames. An FC event can trigger:

- LCD ready interrupt
- Blink (controlling the blink frequency)
- Next state in the Animation State Machine
- Data update if UDCTRL = 01

The Frame Counter is a down counter. It is enabled by writing FCEN in LCD_BACTRL. Optionally, the Frame Counter can be prescaled so that the Frame Counter is decremented at:

- Every frame
- Every second frame
- Every fourth frame
- Every eight frame

This is controlled by the FCPRESC in LCD_BACTRL, see Table 16.9 FCPRESC on page 578

Table 16.9. FCPRESC

FCPRESC	Mode	Description	General equation
00	Div1	$CLK_{FRAME}/1$	$CLK_{FC} = CLK_{FRAME}/2^{FCPRESC}$
01	Div2	$CLK_{FRAME}/2$	
10	Div4	$CLK_{FRAME}/4$	
11	Div8	$CLK_{FRAME}/8$	

The top value for the Frame Counter is set by FCTOP in LCD_BACTRL. Every time the frame counter reaches zero, it is reloaded with the top value, and an event is triggered.

$$CLK_{EVENT} = CLK_{FC}/(1 + FCTOP[5:0]) \text{ Hz}$$

Figure 16.6. LCD Event Frequency Equation

The above equation shows how to set up the LCD event frequency. As an example, if the frame rate is 64Hz, in order to have a LCD event frequency of 0.5Hz, the following parameters should be set accordingly:

- Write FCPRESC to 3 => CLK_{FC} = 8Hz (0.125 seconds)
- Write FCTOP to 15 => CLK_{EVENT} = 0.5Hz (2 seconds)

If higher resolution is required, configure a lower prescaler value and increase the FCTOP value accordingly (e.g. FCPRESC = 2, FCTOP = 31).

16.3.11 LCD Interrupt

The LCD interrupt can be used to synchronize data update. The FC interrupt flag is set at every LCD Frame Counter Event. The interrupt is enabled by setting FC bit in LCD_IEN.

16.3.12 Blink, Blank, and Animation Features

16.3.12.1 Blink

The LCD driver can be configured to blink, alternating all enabled segments between on and off. The blink frequency is given by the CLK_{EVENT} frequency, see 16.3.10 Frame Counter (FC). See 16.3.8 Data Update for details regarding synchronization of the blink feature. The FC must be on for blink to work.

16.3.12.2 Blank

Setting BLANK in LCD_BACTRL will output the “OFF” waveform on all enabled segments, effectively blanking the entire display. Writing the BLANK bit to zero disables the blanking and segment data will be output as normal. See 16.3.8 Data Update for details regarding synchronization of blank.

16.3.12.3 Animation State Machine

The Animation State Machine makes it possible to enable different animations without updating the data registers, allowing specialized patterns running on the LCD panel while the microcontroller remains in Low Energy Mode saving power. The animation feature is available on 8 segments multiplexed with LCD_COM0. The 8 segments can be either segments 0 to 7 or 8 to 15, depending on ALOC in LCD_BACTRL. The animation is implemented as two programmable 8 bit registers that are shifted left or right every other Animation state for a total of 16 states.

The shift operations applied to the shift registers are controlled by AREGASC and AREGBSC in LCD_BACTRL. Note also that the FC must be on for animation to work, as it is the FC event that drives the animation state machine.

The two registers are either OR'ed or AND'ed to achieve the displayed animation pattern. This is controlled by ALOGSEL in LCD_BACTRL. In addition, the regular segment data SEGD0[7:0] / SEGD0[15:8] is OR'ed with the animation pattern to generate the resulting output.

Each state is displayed for one CLK_{EVENT} period, see 16.3.10 Frame Counter (FC). By reading ASTATE in LCD_STATUS, software can identify which state that is currently active in the state sequence. Note that the shifting operation is performed on internal registers that are not accessible in SW (when reading LCD_AREGA and LCD_AREGB, the data that was original written will also be read back). The SW must utilize the knowledge about the current state (ASTATE) to calculate what is currently output. ASTATE is cleared when LCD_AREGA or LCD_AREGB are updated with new values. See Table 16.10 LCD Animation Example on page 580 for an example.

Table 16.10. LCD Animation Example

ASTATE	LCD_AREGA	LCD_AREGB	Resulting Data
0	11000000	11000000	11000000
1	01100000	11000000	11100000
2	01100000	01100000	01100000
3	00110000	01100000	01110000
4	00110000	00110000	00110000
5	00011000	00110000	00111000
6	00011000	00011000	00011000
7	00001100	00011000	00011100
8	00001100	00001100	00001100
9	00000110	00001100	00001110
10	00000110	00000110	00000110
11	00000011	00000110	00000111
12	00000011	00000011	00000011
13	10000001	00000011	10000011
14	10000001	10000001	10000001
15	11000000	10000001	11000001

In the table, AREGASC = SHIFTRIGHT, AREGBSC = SHIFTRIGHT, ALOGSEL = OR and the resulting data is to be displayed on segment lines 7-0 or 15-8 multiplexed with LCD_COM0.

The block diagram of the animation circuit for the LCD is shown in the following figure.

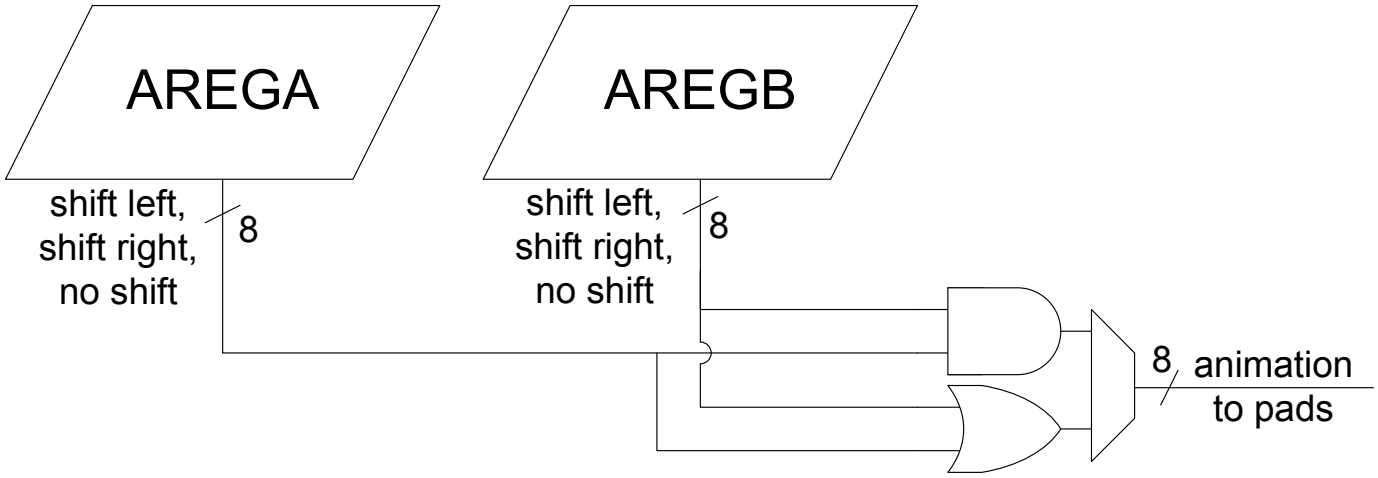


Figure 16.7. LCD Block Diagram of the Animation Circuit

To set up an animation sequence for the LCD, the following sequence is recommended.

1. Write data into the animation registers LCD_AREGA, LCD_AREGB
2. Enable the correct shift direction (if any)
3. Decide which logical function to perform on the registers:
   - ALOGSEL = 0: Data_out = LCD_AREGA & LCD_AREGB
   - ALOGSEL = 1: Data_out = LCD_AREGA | LCD_AREGB
4. Configure the right animation period (CLK_{EVENT})
5. Enable the animation pattern and frame counter (AEN = 1, FCEN = 1)

For updating data in the LCD while it is running an animation, and the new animation data depends on the pattern visible on the LCD, see the following example.

1. Enable the LCD interrupt (the interrupt will be triggered simultaneously as the Animation State machine changes state)
2. In the interrupt handler, read back the current state (ASTATE)
3. Knowing the current state of the Animation State Machine makes it possible to calculate what data that is currently output
4. Modify data as required (Data will be updated at the next Frame Counter Event). It is important that new data is written before the next Frame Counter Event.

16.3.13 LCD in Low Energy Modes

As long as the LFACLK is running (EM0 Active-EM2 DeepSleep), the LCD controller continues to output LCD waveforms according to the data that is currently synchronized to the LCD Driver logic. In addition, the following features are still active if enabled:

- Animation State Machine
- Blink
- LCD Event Interrupt

16.3.14 Register Access

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. Refer to 4.3 Access to Low Energy Peripherals (Asynchronous Registers) for a description on how to perform register accesses to Low Energy Peripherals.

16.3.15 Waveform Examples

The numbers on the illustration's y-axes in the following sections only indicate different voltage levels. All examples are shown with low-power waveforms.

16.3.15.1 Waveforms With Static Bias and Multiplexing

- With static bias and multiplexing, each segment line can be connected to LCD_COM0. When the segment line has the same waveform as LCD_COM0, the LCD panel pixel is turned off, while when the segment line has the opposite waveform, the LCD panel pixel is turned on.
- DC voltage = 0 (over one frame)
- V_{RMS} (on) = V_{LCD}
- V_{RMS} (off) = 0 (V_{SS})

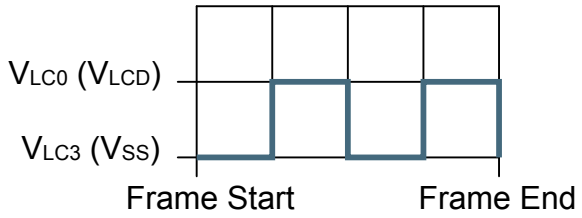


Figure 16.8. LCD Static Bias and Multiplexing - LCD_COM0

16.3.15.2 Waveforms With 1/2 Bias and Duplex Multiplexing

In this mode, each frame is divided into 4 periods. LCD_COM[1:0] lines can be multiplexed with all segment lines. Figures below show 1/2 bias and duplex multiplexing (waveforms show two frames).

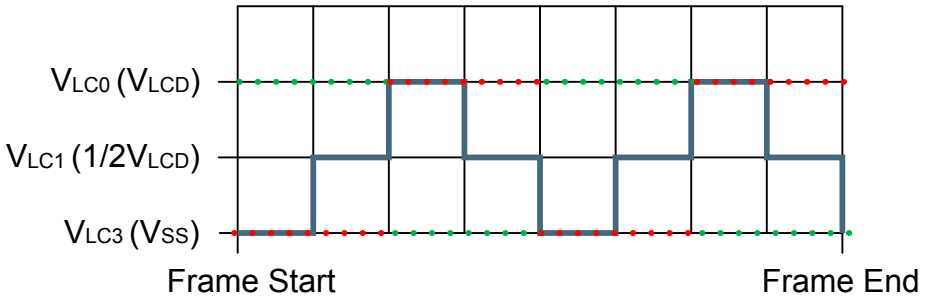


Figure 16.9. LCD 1/2 Bias and Duplex Multiplexing - LCD_COM0

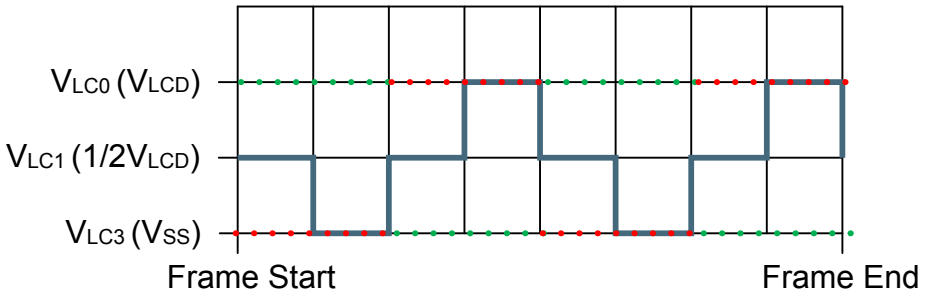


Figure 16.10. LCD 1/2 Bias and Duplex Multiplexing - LCD_COM1

The LCD_SEG0 waveform below is illustrates how different segment waveforms can be multiplexed with the LCD_COM lines in order to turn on and off LCD pixels. As illustrated in the figures below, this waveform will turn ON pixels connected to LCD_COM0, while pixels connected to LCD_COM1 will be turned OFF.

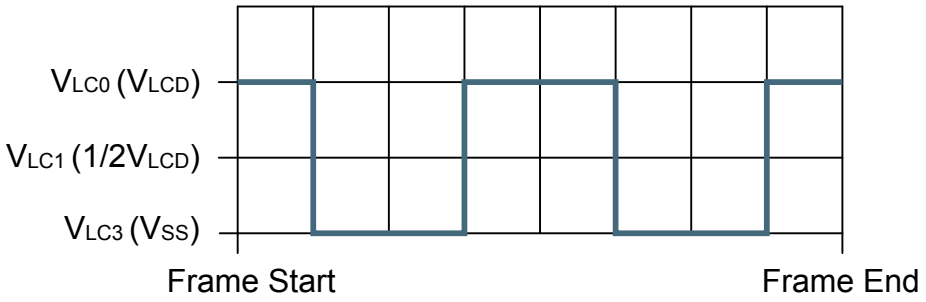


Figure 16.11. LCD 1/2 Bias and Duplex Multiplexing - LCD_SEG0

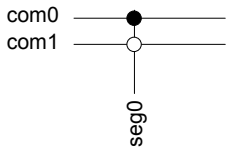


Figure 16.12. LCD 1/2 Bias and Duplex Multiplexing - LCD_SEG0 Connection

The LCD segment between LCD_SEG0 and LCD_COM0 will see the waveform shown in Figure 16.13 LCD 1/2 Bias and Duplex Multiplexing - LCD_SEG0-LCD_COM0 on page 584. In this case, V_{RMS} is 0.79 · V_{LCD}, and the segment is ON.

The LCD segment between LCD_SEG0 and LCD_COM1 will see the waveform shown in Figure 16.14 LCD 1/2 Bias and Duplex Multiplexing - LCD_SEG0-LCD_COM1 on page 585. In this case, V_{RMS} is 0.35 · V_{LCD}, and the segment is OFF.

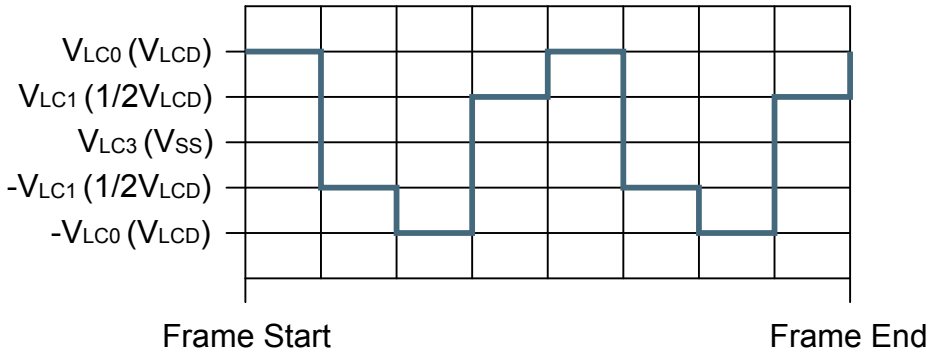


Figure 16.13. LCD 1/2 Bias and Duplex Multiplexing - LCD_SEG0-LCD_COM0

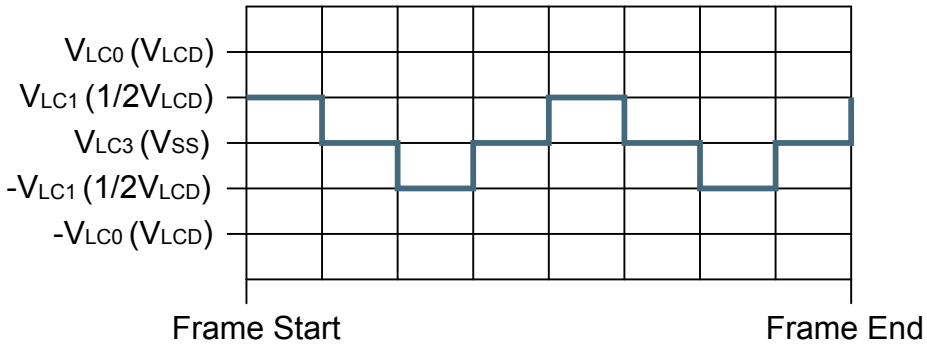


Figure 16.14. LCD 1/2 Bias and Duplex Multiplexing - LCD_SEG0-LCD_COM1

16.3.15.3 Waveforms With 1/3 Bias and Duplex Multiplexing

In this mode, each frame is divided into 4 periods. LCD_COM[1:0] lines can be multiplexed with all segment lines. Figures below show 1/3 bias and duplex multiplexing (waveforms show two frames).

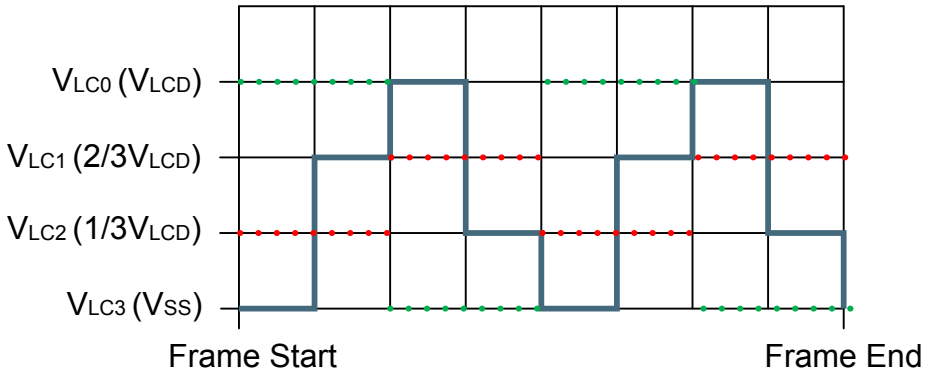


Figure 16.15. LCD 1/3 Bias and Duplex Multiplexing - LCD_COM0

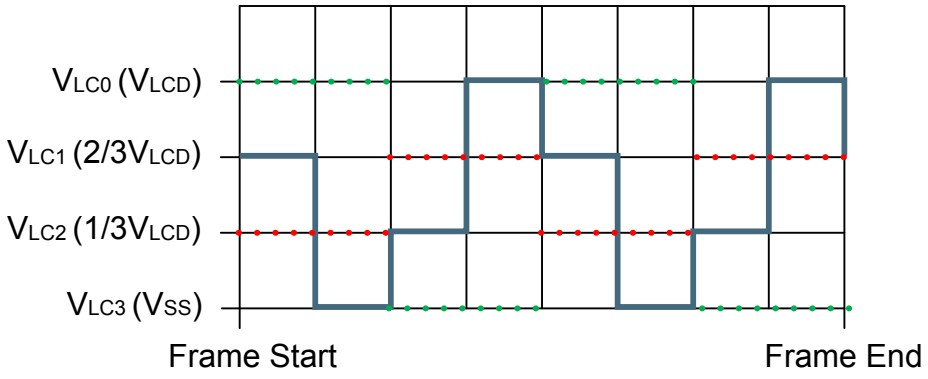


Figure 16.16. LCD 1/3 Bias and Duplex Multiplexing - LCD_COM1

The LCD_SEG0 waveform below illustrates how different segment waveforms can be multiplexed with the COM lines in order to turn on and off LCD pixels. As illustrated in the figures below, this waveform will turn ON pixels connected to LCD_COM0, while pixels connected to LCD_COM1 will be turned OFF.

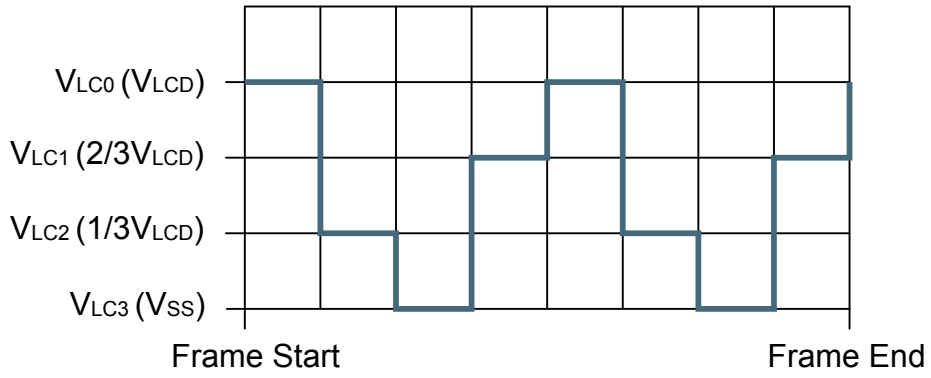


Figure 16.17. LCD 1/3 Bias and Duplex Multiplexing - LCD_SEG0

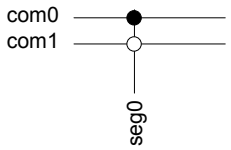


Figure 16.18. LCD 1/3 Bias and Duplex Multiplexing - LCD_SEG0 Connection

The LCD segment between LCD_SEG0 and LCD_COM0 will see the waveform shown in Figure 16.19 LCD 1/3 Bias and Duplex Multiplexing - LCD_SEG0-LCD_COM0 on page 587. In this case, V_{RMS} is 0.75 · V_{LCD}, and the segment is ON.

The LCD segment between LCD_SEG0 and LCD_COM1 will see the waveform shown in Figure 16.20 LCD 1/3 Bias and Duplex Multiplexing - LCD_SEG0-LCD_COM1 on page 588. In this case, V_{RMS} is 0.33 · V_{LCD}, and the segment is OFF.

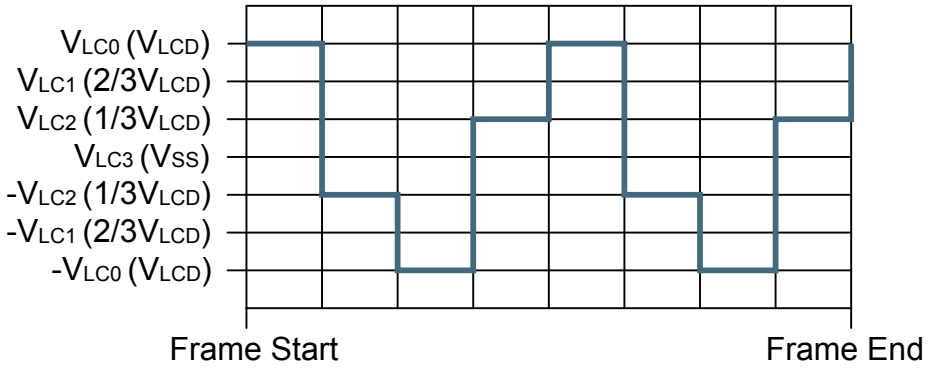


Figure 16.19. LCD 1/3 Bias and Duplex Multiplexing - LCD_SEG0-LCD_COM0

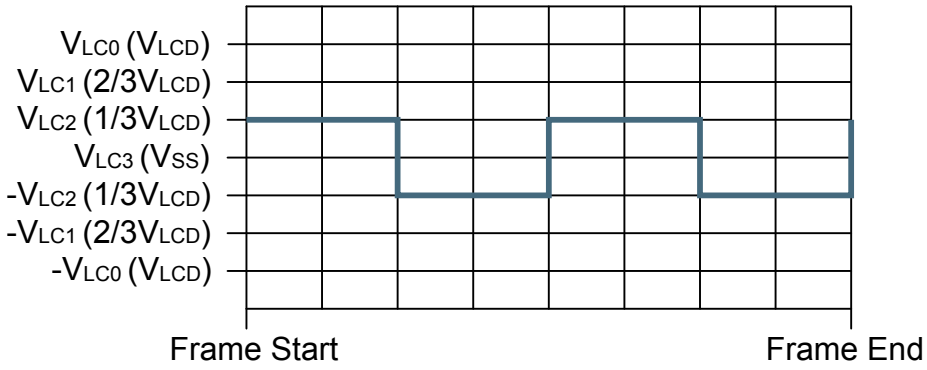


Figure 16.20. LCD 1/3 Bias and Duplex Multiplexing - LCD_SEG0-LCD_COM1

16.3.15.4 Waveforms With 1/2 Bias and Triplex Multiplexing

In this mode, each frame is divided into 6 periods. LCD_COM[2:0] lines can be multiplexed with all segment lines. Figures below show 1/2 bias and triplex multiplexing (waveforms show two frames).

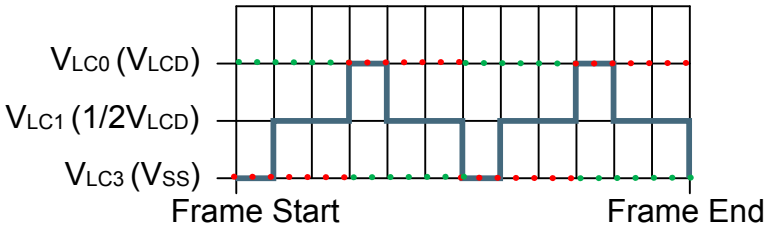


Figure 16.21. LCD 1/2 Bias and Triplex Multiplexing - LCD_COM0

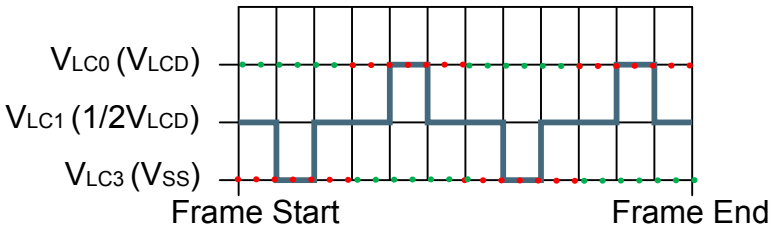


Figure 16.22. LCD 1/2 Bias and Triplex Multiplexing - LCD_COM1

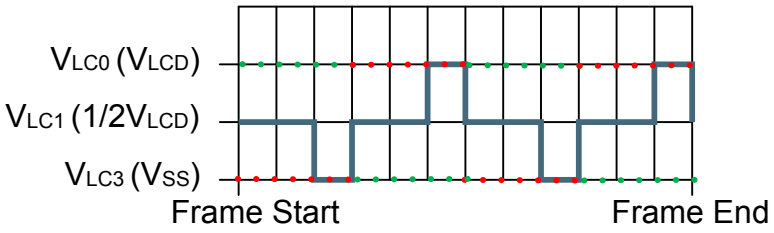


Figure 16.23. LCD 1/2 Bias and Triplex Multiplexing - LCD_COM2

The LCD_SEG0 waveform below illustrates how different segment waveforms can be multiplexed with the COM lines in order to turn on and off LCD pixels. As illustrated in the figures below, this waveform will turn ON pixels connected to LCD_COM1, while pixels connected to LCD_COM0 and LCD_COM2 will be turned OFF.

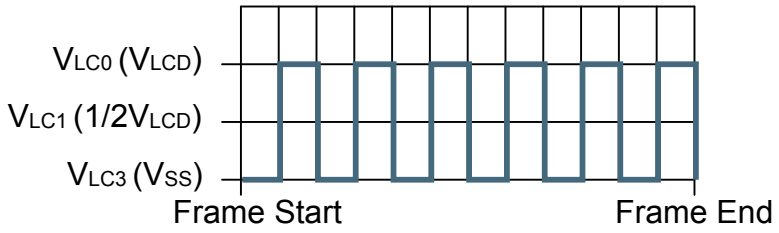


Figure 16.24. LCD 1/2 Bias and Triplex Multiplexing - LCD_SEG0

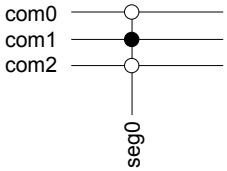


Figure 16.25. LCD 1/2 Bias and Triplex Multiplexing - LCD_SEG0 Connection

The LCD segment between LCD_SEG0 and LCD_COM0 will see the waveform shown in Figure 16.26 LCD 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM0 on page 590. In this case, V_{RMS} is 0.4 · V_{LCD}, and the segment is OFF.

The LCD segment between LCD_SEG0 and LCD_COM1 will see the waveform shown in Figure 16.27 LCD 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM1 on page 591. In this case, V_{RMS} is 0.7 · V_{LCD}, and the segment is ON.

The LCD segment between LCD_SEG0 and LCD_COM2 will see the waveform shown in Figure 16.28 LCD 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM2 on page 591. In this case, V_{RMS} is 0.4 · V_{LCD}, and the segment is OFF.

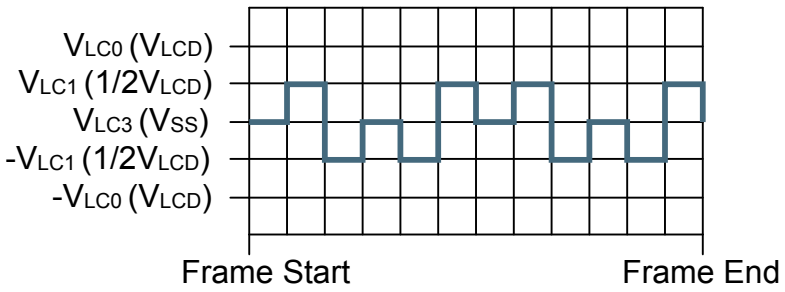


Figure 16.26. LCD 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM0

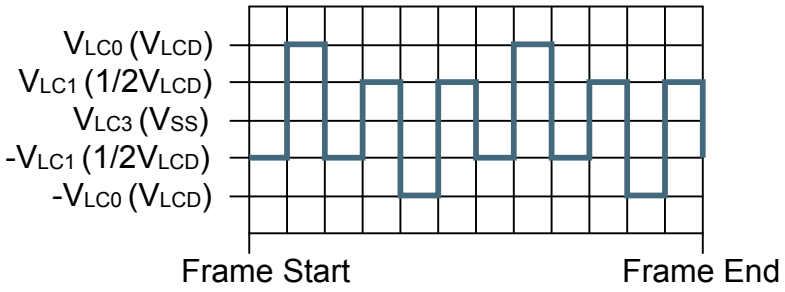


Figure 16.27. LCD 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM1

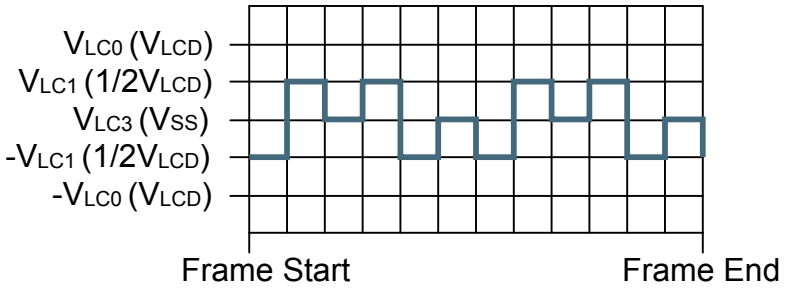


Figure 16.28. LCD 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM2

16.3.15.5 Waveforms With 1/3 Bias and Triplex Multiplexing

In this mode, each frame is divided into 6 periods. LCD_COM[2:0] lines can be multiplexed with all segment lines. Figures below show 1/3 bias and triplex multiplexing (waveforms show two frames).

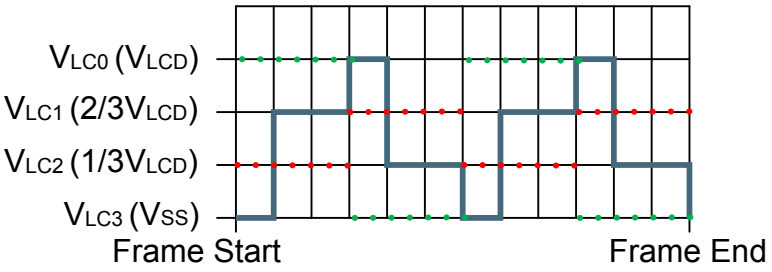


Figure 16.29. LCD 1/3 Bias and Triplex Multiplexing - LCD_COM0

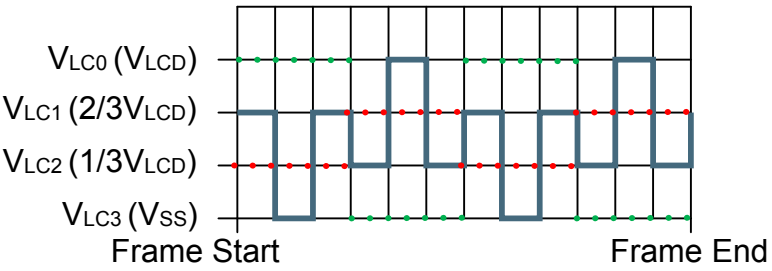


Figure 16.30. LCD 1/3 Bias and Triplex Multiplexing - LCD_COM1

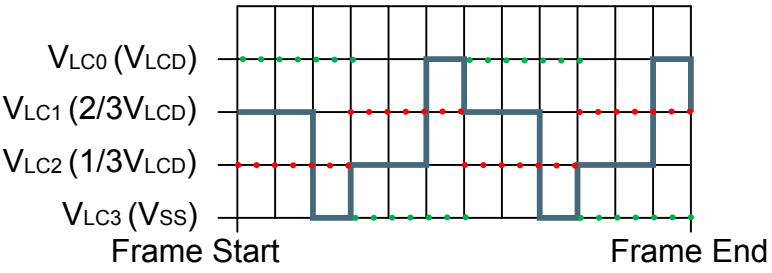


Figure 16.31. LCD 1/3 Bias and Triplex Multiplexing - LCD_COM2

The LCD_SEG0 waveform illustrates how different segment waveforms can be multiplexed with the COM lines in order to turn on and off LCD pixels. As illustrated in the figures below, this waveform will turn ON pixels connected to LCD_COM1, while pixels connected to LCD_COM0 and LCD_COM2 will be turned OFF.

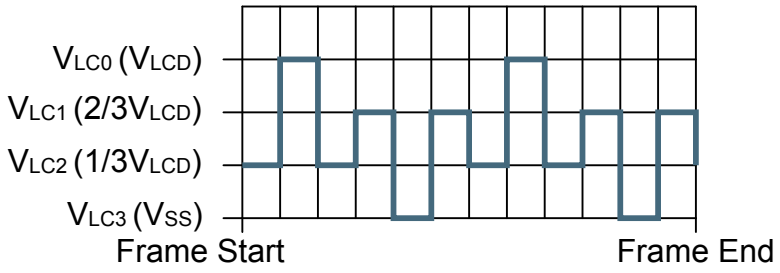


Figure 16.32. LCD 1/3 Bias and Triplex Multiplexing - LCD_SEG0

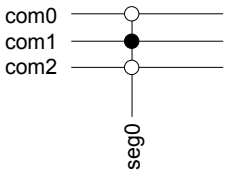


Figure 16.33. LCD 1/3 Bias and Triplex Multiplexing - LCD_SEG0 Connection

The LCD segment between LCD_SEG0 and LCD_COM0 will see the waveform shown in Figure 16.34 LCD 1/3 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM0 on page 593. In this case, V_{RMS} is 0.33 · V_{LCD}, and the segment is OFF.

The LCD segment between LCD_SEG0 and LCD_COM1 will see the waveform shown in Figure 16.35 LCD 1/3 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM1 on page 594. In this case, V_{RMS} is 0.64 · V_{LCD}, and the segment is ON.

The LCD segment between LCD_SEG0 and LCD_COM2 will see the waveform shown in Figure 16.36 LCD 1/3 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM2 on page 594. In this case, V_{RMS} is 0.33 · V_{LCD}, and the segment is OFF.

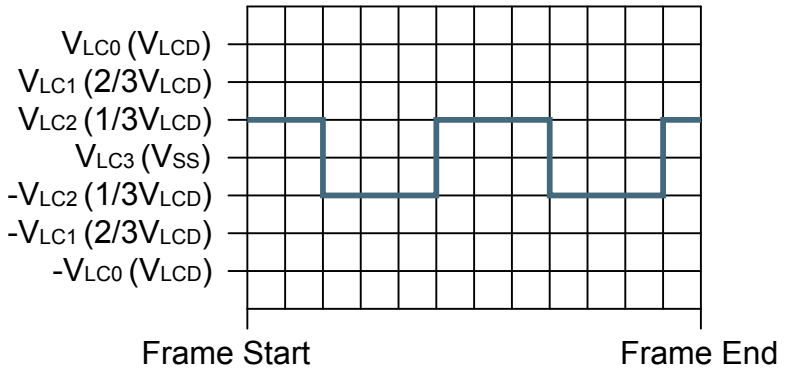


Figure 16.34. LCD 1/3 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM0

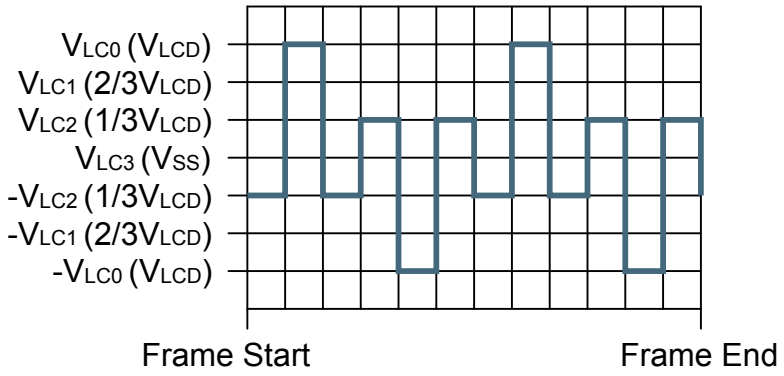


Figure 16.35. LCD 1/3 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM1

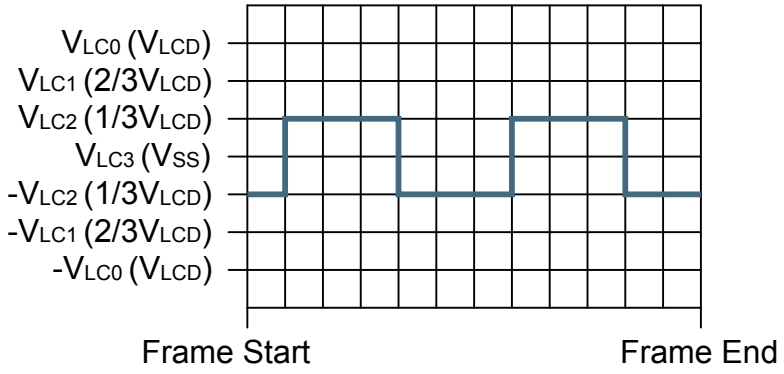


Figure 16.36. LCD 1/3 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM2

16.3.15.6 Waveforms With 1/3 Bias and Quadruplex Multiplexing

In this mode, each frame is divided into 8 periods. All COM lines can be multiplexed with all segment lines. Figures below show 1/3 bias and quadruplex multiplexing (waveforms show two frames).

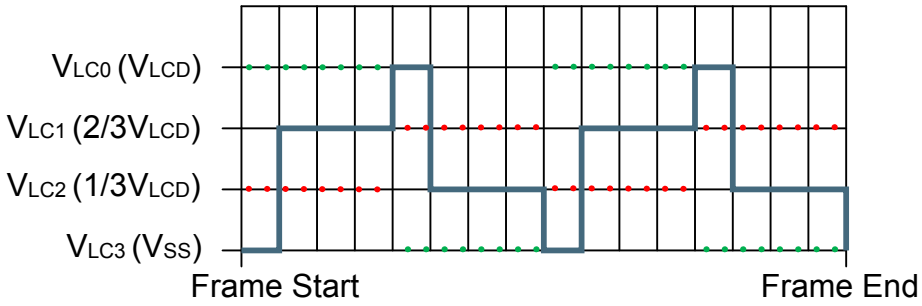


Figure 16.37. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_COM0

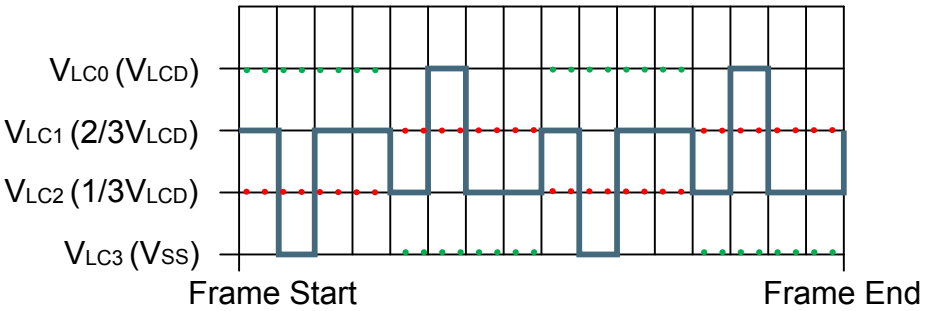


Figure 16.38. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_COM1

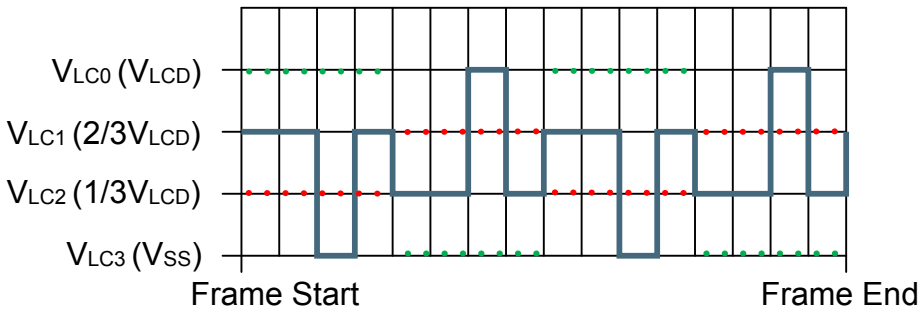


Figure 16.39. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_COM2

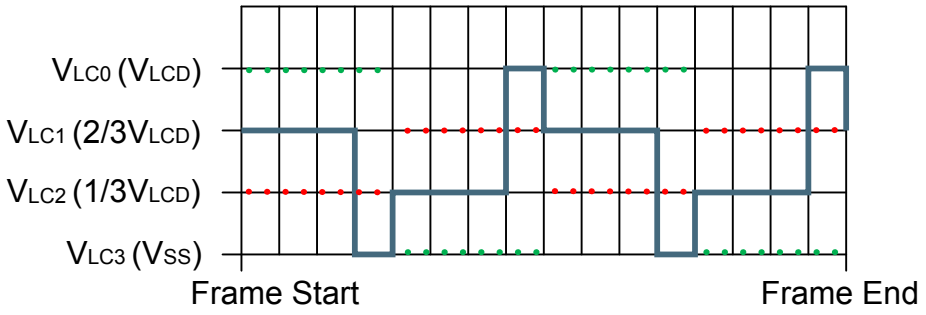


Figure 16.40. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_COM3

The LCD_SEG0 waveform below illustrates how different segment waveforms can be multiplexed with the COM lines in order to turn on and off LCD pixels. As illustrated in the figures below, this waveform will turn ON pixels connected to LCD_COM0 and LCD_COM2, while pixels connected to LCD_COM1 and LCD_COM3 will be turned OFF.

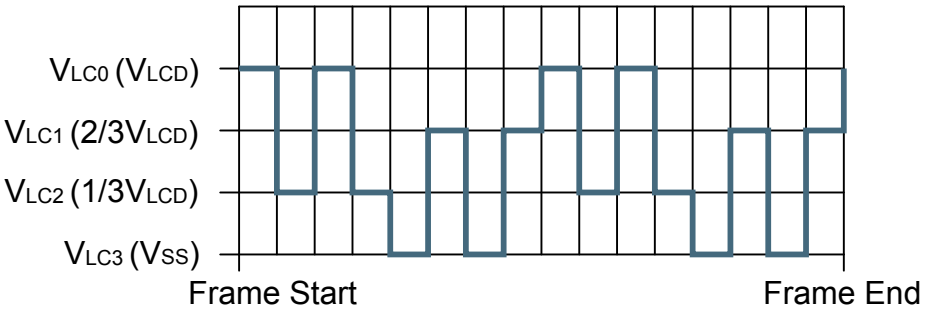


Figure 16.41. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_SEG0

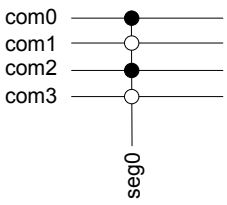


Figure 16.42. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_SEG0 Connection

The LCD segment between LCD_SEG0 and LCD_COM0 will see the waveform shown in Figure 16.43 LCD 1/3 Bias and Quadruplex Multiplexing - LCD_SEG0-LCD_COM0 on page 597. In this case, V_{RMS} is 0.58 · V_{LCD}, and the segment is ON.

The LCD segment between LCD_SEG0 and LCD_COM1 will see the waveform shown in Figure 16.44 LCD 1/3 Bias and Quadruplex Multiplexing - LCD_SEG0-LCD_COM1 on page 597. In this case, V_{RMS} is 0.33 · V_{LCD}, and the segment is OF.

The LCD segment between LCD_SEG0 and LCD_COM2 will see the waveform shown in Figure 16.45 LCD 1/3 Bias and Quadruplex Multiplexing - LCD_SEG0-LCD_COM2 on page 598. In this case, V_{RMS} is 0.58 · V_{LCD}, and the segment is ON.

The LCD segment between LCD_SEG0 and LCD_COM3 will see the waveform shown in Figure 16.46 LCD 1/3 Bias and Quadruplex Multiplexing- LCD_SEG0-LCD_COM3 on page 598. In this case, V_{RMS} is 0.33 · V_{LCD}, and the segment is OFF.

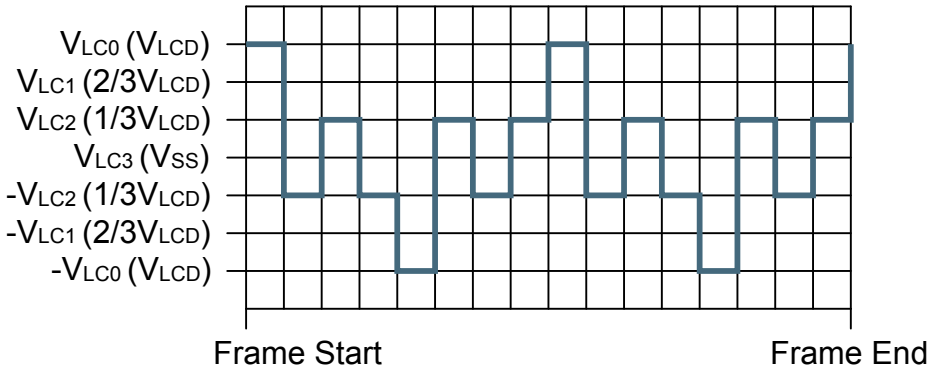


Figure 16.43. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_SEG0-LCD_COM0

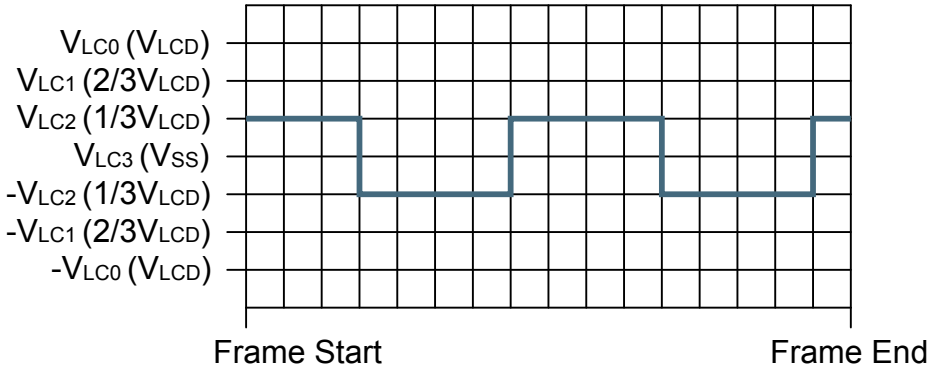


Figure 16.44. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_SEG0-LCD_COM1

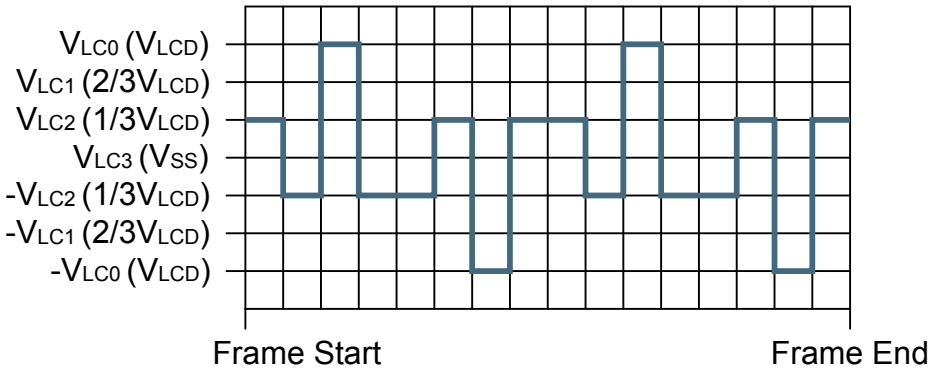


Figure 16.45. LCD 1/3 Bias and Quadruplex Multiplexing - LCD_SEG0-LCD_COM2

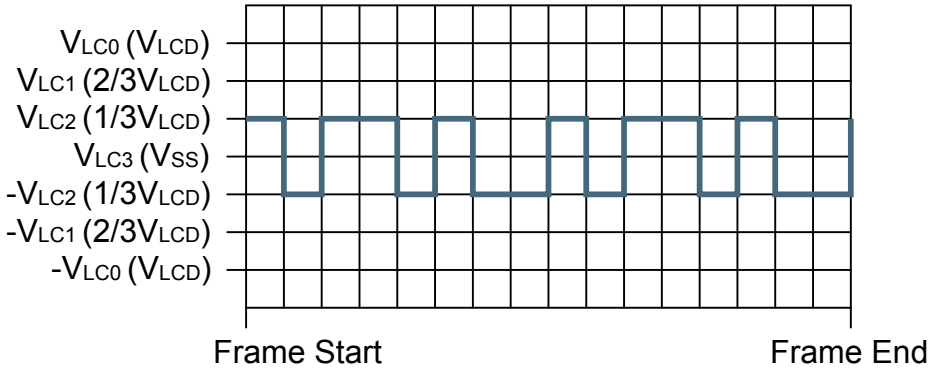


Figure 16.46. LCD 1/3 Bias and Quadruplex Multiplexing- LCD_SEG0-LCD_COM3

16.3.15.7 Waveforms With Charge Redistribution

This example assumes a 32.768 kHz clock prescaled in the CMU by 2, triplex multiplexing, and an FRDIV of 90 which gives 30 frames per second. The charge redistribution is 1% of each phase. The normal power waveform is shown with segment 0 data of {1,1,0}.

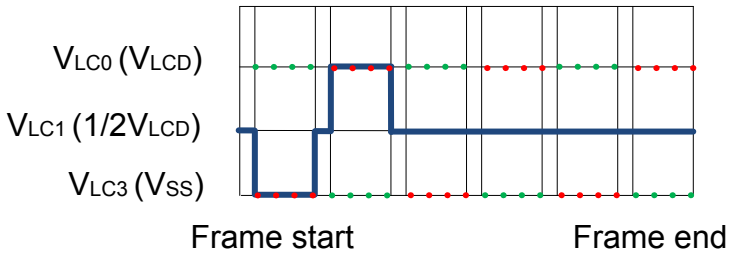


Figure 16.47. LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_COM0

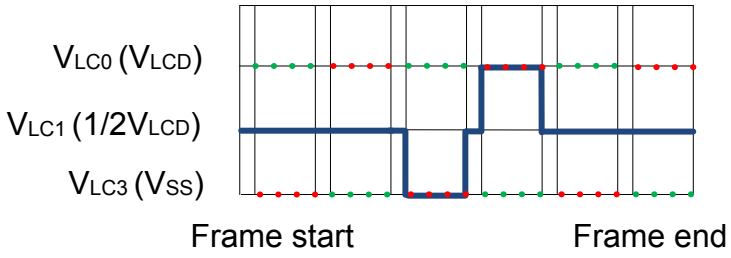


Figure 16.48. LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_COM1

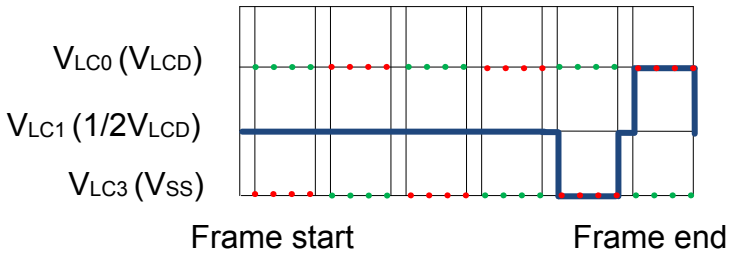


Figure 16.49. LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_COM2

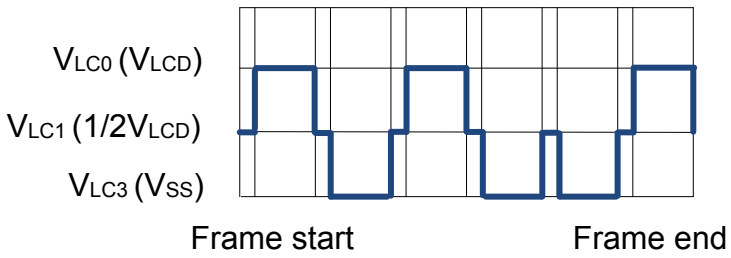


Figure 16.50. LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_SEG0

The LCD segment between LCD_SEG0 and LCD_COM0 will see the waveform shown in Figure 16.51 LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM0 on page 600. In this case, V_{RMS} is 0.7 · V_{LCD}, and the segment is ON.

The LCD segment between LCD_SEG0 and LCD_COM1 will see the waveform shown in Figure 16.52 LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM1 on page 600. In this case, V_{RMS} is 0.7 · V_{LCD}, and the segment is ON.

The LCD segment between LCD_SEG0 and LCD_COM2 will see the waveform shown in Figure 16.53 LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM2 on page 601. In this case, V_{RMS} is 0.4 · V_{LCD}, and the segment is OFF.

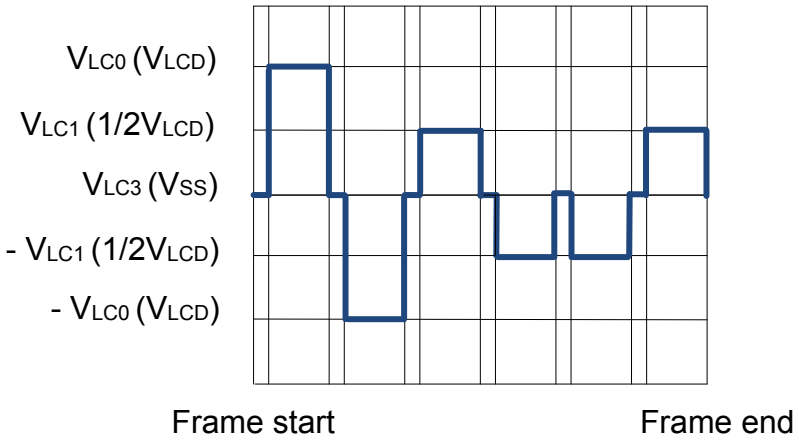


Figure 16.51. LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM0

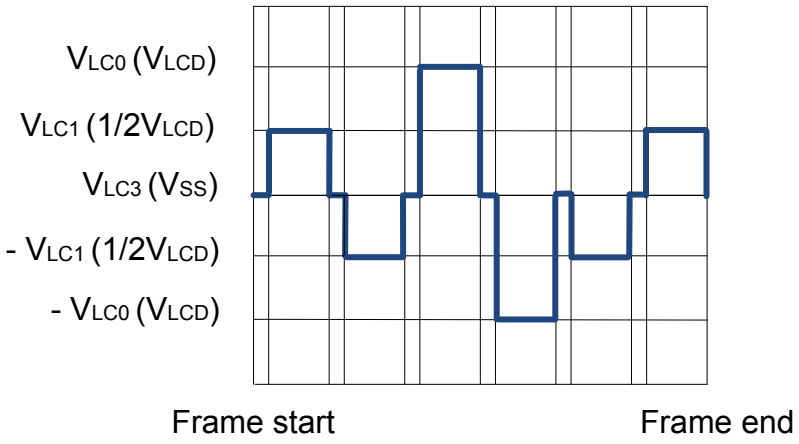


Figure 16.52. LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM1

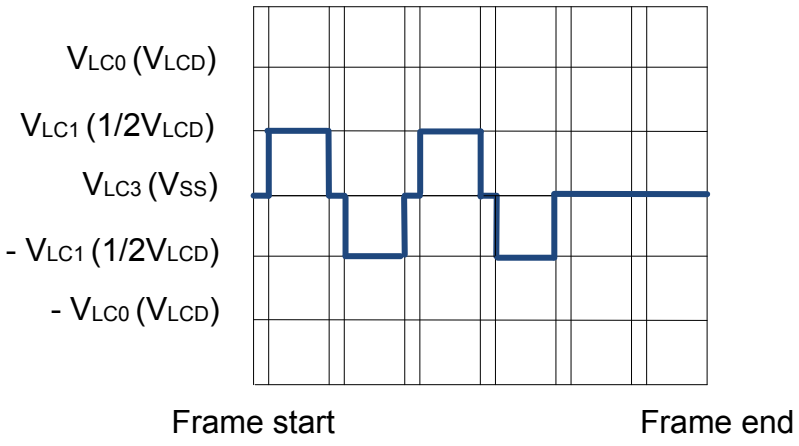


Figure 16.53. LCD Charge Redist - 1/2 Bias and Triplex Multiplexing - LCD_SEG0-LCD_COM2

16.4 Register Map

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

Offset	Name	Type	Description
0x000	LCD_CTRL	RW	Control Register
0x004	LCD_DISPCTRL	RW	Display Control Register
0x008	LCD_SEGEN	RW	Segment Enable Register
0x00C	LCD_BACTRL	RW	Blink and Animation Control Register
0x010	LCD_STATUS	R	Status Register
0x014	LCD_AREGA	RW	Animation Register a
0x018	LCD_AREGB	RW	Animation Register B
0x01C	LCD_IF	R	Interrupt Flag Register
0x020	LCD_IFS	W1	Interrupt Flag Set Register
0x024	LCD_IFC	(R)W1	Interrupt Flag Clear Register
0x028	LCD_IEN	RW	Interrupt Enable Register
0x030	LCD_BIASCTRL	RW	Analog BIAS Control
0x040	LCD_SEGD0L	RW	Segment Data Low Register 0
0x044	LCD_SEGD1L	RW	Segment Data Low Register 1
0x048	LCD_SEGD2L	RW	Segment Data Low Register 2
0x04C	LCD_SEGD3L	RW	Segment Data Low Register 3
0x050	LCD_SEGD0H	RW	Segment Data High Register 0
0x054	LCD_SEGD1H	RW	Segment Data High Register 1
0x058	LCD_SEGD2H	RW	Segment Data High Register 2
0x05C	LCD_SEGD3H	RW	Segment Data High Register 3
0x060	LCD_SEGD4L	RW	Segment Data Low Register 4
0x064	LCD_SEGD5L	RW	Segment Data Low Register 5
0x068	LCD_SEGD6L	RW	Segment Data Low Register 6
0x06C	LCD_SEGD7L	RW	Segment Data Low Register 7
0x070	LCD_SEGD4H	RW	Segment Data High Register 4
0x074	LCD_SEGD5H	RW	Segment Data High Register 5
0x078	LCD_SEGD6H	RW	Segment Data High Register 6
0x07C	LCD_SEGD7H	RW	Segment Data High Register 7
0x0C0	LCD_FREEZE	RW	Freeze Register
0x0C4	LCD_SYNCBUSY	R	Synchronization Busy Register
0x0F0	LCD_FRAMERATE	RW	Frame Rate
0x0F4	LCD_SEGEN2	RW	Segment Enable (32 to 39)

16.5 Register Description

16.5.1 LCD_CTRL - Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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									0																					0x0		0
Access									RW																					RW		RW
Name									DSC																					UDCTRL		EN

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	DSC	0	RW	**Direct Segment Control**

This bit enables direct control over bias levels for each SEG/COM line.

Value	Description
0	DSC disable
1	DSC enable

Bit	Name	Reset	Access	Description
22:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:1	UDCTRL	0x0	RW	**Update Data Control**

These bits control how data from the SEGDn registers are transferred to the LCD driver.

Value	Mode	Description
0	REGULAR	The data transfer is controlled by SW. Transfer is performed as soon as possible
1	FCEVENT	The data transfer is done at the next event triggered by the Frame Counter
2	FRAMESTART	The data transfer is done continuously at every LCD frame start

Bit	Name	Reset	Access	Description
0	EN	0	RW	**LCD Enable**

When this bit is set, the LCD driver is enabled and the driver will start outputting waveforms on the com/segment lines.

16.5.2 LCD_DISPCTRL - Display 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									0x3F								0		0x0		
Access			RW				RW			RW									RW								RW		RW		
Name			MODE				BIAS			CHGRDST									CONTRAST								WAVE		MUX		

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	MODE	0x0	RW	**Mode Setting**

This field determines the LCD mode of operation.

Value	Mode	Description
0	NOEXTCAP	No External Cap. Uses an internal current source to generate VLCD. Use CONTRAST[4:0] to control VLCD.
1	STEPDOWN	Use step down control with VLCD less than VDD. Use CONTRAST[5:0] to control VLCD level, and use SPEED to adjust VLCD drive strength.
2	CPINTOSC	Charge pump used with internal oscillator. Use CONTRAST[5:0] to control VLCD level, and use SPEED to adjust oscillator frequency.

27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:24	BIAS	0x0	RW	**Bias Configuration**

These bits set the bias mode for the LCD Driver.

Value	Mode	Description
0	STATIC	Static
1	ONEHALF	1/2 Bias
2	ONETHIRD	1/3 Bias
3	ONEFOURTH	1/4 Bias

23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:20	CHGRDST	0x1	RW	**Charge Redistribution Cycles**

Selects number of prescaled low frequency clock cycles for charge redistribution.

Value	Mode	Description
0	DISABLE	Disable charge redistribution.
1	ONE	Use 1 prescaled low frequency clock cycle for charge redistribution.
2	TWO	Use 2 prescaled low frequency clock cycles for charge redistribution.

Bit	Name	Reset	Access	Description
	3	THREE		Use 3 prescaled low frequency clock cycles for charge redistribution.
	4	FOUR		Use 4 prescaled low frequency clock cycles for charge redistribution.
19:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CONTRAST	0x3F	RW	**Contrast Control**
	This controls the VLCD supply voltage.			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	WAVE	0	RW	**Waveform Selection**
	This bit configures the output waveform.			
	Value	Mode		Description
	0	LOWPOWER		Low power waveform
	1	NORMAL		Normal waveform
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	MUX	0x0	RW	**Mux Configuration**
	These bits set the multiplexing mode for the LCD Driver.			
	Value	Mode		Description
	0	STATIC		Static
	1	DUPLEX		Duplex
	2	TRIPLEX		Triplex
	3	QUADRUPLEX		Quadruplex
	5	SEXTAPLEX		Sextaplex
	7	OCTAPLEX		Octaplex

16.5.3 LCD_SEGEN - Segment Enable 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	0x00000000																															
Access	RW																															
Name	SEGEN																															

Bit	Name	Reset	Access	Description
31:0	SEGEN	0x00000000	RW	**Segment Enable**
	Determines which segment lines are enabled from (0 to 31). The GPIO pin also needs to be configured as DISABLED in the GPIO pin configuration.			

16.5.4 LCD_BACTRL - Blink and Animation Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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				0					0x00						0x0									0	0	0x0		0x0		0	0	0
Access				RW					RW						RW									RW	RW	RW		RW		RW	RW	RW
Name				ALOC					FCTOP						FCPRESC									FCEN	ALOGSEL	AREGBSC		AREGASC		AEN	BLANK	BLINKEN

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	ALOC	0	RW	**Animation Location**

Set the LCD segments which animation applies to

Value	Mode	Description
0	SEG0TO7	Animation appears on segments 0 to 7
1	SEG8TO15	Animation appears on segments 8 to 15

27:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:18	FCTOP	0x00	RW	**Frame Counter Top Value**

These bits contain the Top Value for the Frame Counter: $CLK_{EVENT} = CLK_{FC} / (1 + FCTOP[5:0])$.

17:16	FCPRESC	0x0	RW	**Frame Counter Prescaler**

These bits controls the prescaling value for the Frame Counter input clock.

Value	Mode	Description
0	DIV1	$CLK_{FC} = CLK_{FRAME} / 1$
1	DIV2	$CLK_{FC} = CLK_{FRAME} / 2$
2	DIV4	$CLK_{FC} = CLK_{FRAME} / 4$
3	DIV8	$CLK_{FC} = CLK_{FRAME} / 8$

15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	FCEN	0	RW	**Frame Counter Enable**

When this bit is set, the frame counter is enabled.

7	ALOGSEL	0	RW	**Animate Logic Function Select**

When this bit is set, the animation registers are AND'ed together. When this bit is cleared, the animation registers are OR'ed together.

Value	Mode	Description
0	AND	AREGA and AREGB AND'ed

Bit	Name	Reset	Access	Description
	1	OR		AREGA and AREGB OR'ed
6:5	AREGBSC	0x0	RW	**Animate Register B Shift Control**

These bits controls the shift operation that is performed on Animation register B.

Value	Mode	Description
0	NOSHIFT	No Shift operation on Animation Register B
1	SHIFTLEFT	Animation Register B is shifted left
2	SHIFTRIGHT	Animation Register B is shifted right

4:3	AREGASC	0x0	RW	**Animate Register a Shift Control**

These bits controls the shift operation that is performed on Animation register A.

Value	Mode	Description
0	NOSHIFT	No Shift operation on Animation Register A
1	SHIFTLEFT	Animation Register A is shifted left
2	SHIFTRIGHT	Animation Register A is shifted right

2	AEN	0	RW	**Animation Enable**

When this bit is set, the animate function is enabled.

1	BLANK	0	RW	**Blank Display**

When this bit is set, all segment output waveforms are configured to blank the LCD display. The Segment Data Registers are not affected when writing this bit.

Value	Description
0	Display is not "blanked"
1	Display is "blanked"

0	BLINKEN	0	RW	**Blink Enable**

When this bit is set, the Blink function is enabled. Every "ON" segment will alternate between on and off at every Frame Counter Event.

16.5.5 LCD_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																								0					0x0			
Access																								R					R			
Name																								BLINK					ASTATE			

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	BLINK	0	R	**Blink State**
	This bits indicates the blink status. If this bit is 1, all segments are off. If this bit is 0, the segments(LCD_SEGDxn) which are set to 1 are on.			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	ASTATE	0x0	R	**Current Animation State**
	Contains the current animation state (0-15).			

16.5.6 LCD_AREGA - Animation Register a (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																												0x00				
Access																												RW				
Name																												AREGA				

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	AREGA	0x00	RW	**Animation Register a Data**
	This register contains the A data for generating animation pattern.			

16.5.7 LCD_AREGB - Animation Register B (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									AREGB							

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	AREGB	0x00	RW	**Animation Register B Data**
	This register contains the B data for generating animation pattern.			

16.5.8 LCD_IF - Interrupt Flag 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																																0
Access																																R
Name																																FC

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	FC	0	R	**Frame Counter Interrupt Flag**
	Set when Frame Counter is zero.			

16.5.9 LCD_IFS - Interrupt Flag Set 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																																0
Access																																W1
Name																																FC

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	FC	0	W1	**Frame Counter Interrupt Flag Set**
	Write to 1 to set FC interrupt flag.			

16.5.10 LCD_IFC - Interrupt Flag Clear 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																																0
Access																																(R)W1
Name																																FC

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	FC	0	(R)W1	**Frame Counter Interrupt Flag Clear**
	Write to 1 to clear FC interrupt flag.			

16.5.11 LCD_IEN - Interrupt Enable 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																																0
Access																																RW
Name																																FC

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	FC	0	RW	**Frame Counter Interrupt Enable**
	Set to enable interrupt on frame counter interrupt flag.			

16.5.12 LCD_BIASCTRL - Analog BIAS 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					0x0					0x0		
Access																				RW					RW					RW		
Name																				BUFBIAS					BUFDRV					SPEED		

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*		
12:10	BUFBIAS	0x0	RW	**Buffer Bias Setting**
	This field sets the operating bias current for the buffers.			
9:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:4	BUFDRV	0x0	RW	**Buffer Drive Strength**
	This field is used to set the buffer driver strength.			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	SPEED	0x0	RW	**SPEED Adjustment**
	This field is used in mode 1 to adjust the drive strength to the resistor string. For mode 3 this field is used to adjust the speed of the internal oscillator.			

16.5.13 LCD_SEGD0L - Segment Data Low Register 0 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x00000000																															
Access	RW																															
Name	SEGD0L																															

Bit	Name	Reset	Access	Description
31:0	SEGD0L	0x00000000	RW	**COM0 Segment Data Low**
	This register contains segment data for segment lines 0-31 for COM0.			

16.5.14 LCD_SEGD1L - Segment Data Low Register 1 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x00000000																															
Access	RW																															
Name	SEGD1L																															

Bit	Name	Reset	Access	Description
31:0	SEGD1L	0x00000000	RW	**COM1 Segment Data Low**
	This register contains segment data for segment lines 0-31 for COM1.			

16.5.15 LCD_SEGD2L - Segment Data Low Register 2 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x00000000																															
Access	RW																															
Name	SEGD2L																															

Bit	Name	Reset	Access	Description
31:0	SEGD2L	0x00000000	RW	**COM2 Segment Data Low**
	This register contains segment data for segment lines 0-31 for COM2.			

16.5.16 LCD_SEGD3L - Segment Data Low Register 3 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x00000000																															
Access	RW																															
Name	SEGD3L																															

Bit	Name	Reset	Access	Description
31:0	SEGD3L	0x00000000	RW	**COM3 Segment Data Low**
	This register contains segment data for segment lines 0-31 for COM3.			

16.5.17 LCD_SEGD0H - Segment Data High Register 0 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									SEGD0H							

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	SEGD0H	0x00	RW	**COM0 Segment Data High**
	This register contains segment data for segment lines 32-39 for COM0.			

16.5.18 LCD_SEGD1H - Segment Data High Register 1 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									SEGD1H							

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	SEGD1H	0x00	RW	**COM1 Segment Data High**
	This register contains segment data for segment lines 32-39 for COM1.			

16.5.19 LCD_SEGD2H - Segment Data High Register 2 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									SEGD2H							

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	SEGD2H	0x00	RW	**COM2 Segment Data High**
	This register contains segment data for segment lines 32-39 for COM2.			

16.5.20 LCD_SEGD3H - Segment Data High Register 3 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									SEGD3H							

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	SEGD3H	0x00	RW	**COM3 Segment Data High**
	This register contains segment data for segment lines 32-39 for COM3.			

16.5.21 LCD_SEGD4L - Segment Data Low Register 4 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																0x00000000																
Access																RW																
Name																SEGD4L																

Bit	Name	Reset	Access	Description
31:0	SEGD4L	0x00000000	RW	**COM4 Segment Data**
	This register contains segment data for segment lines 0-31 for COM4.			

16.5.22 LCD_SEGD5L - Segment Data Low Register 5 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x00000000																															
Access	RW																															
Name	SEGD5L																															

Bit	Name	Reset	Access	Description
31:0	SEGD5L	0x00000000	RW	**COM5 Segment Data**
	This register contains segment data for segment lines 0-31 for COM5.			

16.5.23 LCD_SEGD6L - Segment Data Low Register 6 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x00000000																															
Access	RW																															
Name	SEGD6L																															

Bit	Name	Reset	Access	Description
31:0	SEGD6L	0x00000000	RW	**COM6 Segment Data**
	This register contains segment data for segment lines 0-31 for COM6.			

16.5.24 LCD_SEGD7L - Segment Data Low Register 7 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x00000000																															
Access	RW																															
Name	SEGD7L																															

Bit	Name	Reset	Access	Description
31:0	SEGD7L	0x00000000	RW	**COM7 Segment Data**
	This register contains segment data for segment lines 0-31 for COM7.			

16.5.25 LCD_SEGD4H - Segment Data High Register 4 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									SEGD4H							

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	SEGD4H	0x00	RW	**COM0 Segment Data High**
	This register contains segment data for segment lines 32-39 for COM4.			

16.5.26 LCD_SEGD5H - Segment Data High Register 5 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									SEGD5H							

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	SEGD5H	0x00	RW	**COM1 Segment Data High**
	This register contains segment data for segment lines 32-39 for COM5.			

16.5.27 LCD_SEGD6H - Segment Data High Register 6 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									SEGD6H							

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	SEGD6H	0x00	RW	**COM2 Segment Data High**
	This register contains segment data for segment lines 32-39 for COM6.			

16.5.28 LCD_SEGD7H - Segment Data High Register 7 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RW							
Name																									SEGD7H							

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	SEGD7H	0x00	RW	**COM3 Segment Data High**
	This register contains segment data for segment lines 32-39 for COM7.			

16.5.29 LCD_FREEZE - Freeze Register

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																															0	0
Access																															RW	RW
Name																															LCDGATE	REGFREEZE

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	LCDGATE	0	RW	**LCD Gate**

Tristate the LCD pins. The gating or un-gating occurs on Frame boundaries.

Value	Mode	Description
0	UNGATE	LCD BIAS voltages driven onto pins.
1	GATE	LCD BIAS MUX tristated at the pins.

Bit	Name	Reset	Access	Description
0	REGFREEZE	0	RW	**Register Update Freeze**

When set, the update of the LCD is postponed until this bit is cleared. Use this bit to update several registers simultaneously.

Value	Mode	Description
0	UPDATE	Each write access to an LCD register is updated into the Low Frequency domain as soon as possible.
1	FREEZE	The LCD is not updated with the new written value.

16.5.30 LCD_SYNCBUSY - Synchronization Busy Register

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													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access													R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name													SEGD7H	SEGD6H	SEGD5H	SEGD4H	SEGD7L	SEGD6L	SEGD5L	SEGD4L	SEGD3H	SEGD2H	SEGD1H	SEGD0H	SEGD3L	SEGD2L	SEGD1L	SEGD0L	AREGB	AREGA	BACTRL	CTRL

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	SEGD7H	0	R	**SEGD7H Register Busy**
	Set when the value written to SEGD7H is being synchronized.			
18	SEGD6H	0	R	**SEGD6H Register Busy**
	Set when the value written to SEGD6H is being synchronized.			
17	SEGD5H	0	R	**SEGD5H Register Busy**
	Set when the value written to SEGD5H is being synchronized.			
16	SEGD4H	0	R	**SEGD4H Register Busy**
	Set when the value written to SEGD4H is being synchronized.			
15	SEGD7L	0	R	**SEGD7L Register Busy**
	Set when the value written to SEGD7L is being synchronized.			
14	SEGD6L	0	R	**SEGD6L Register Busy**
	Set when the value written to SEGD6L is being synchronized.			
13	SEGD5L	0	R	**SEGD5L Register Busy**
	Set when the value written to SEGD5L is being synchronized.			
12	SEGD4L	0	R	**SEGD4L Register Busy**
	Set when the value written to SEGD4L is being synchronized.			
11	SEGD3H	0	R	**SEGD3H Register Busy**
	Set when the value written to SEGD3H is being synchronized.			
10	SEGD2H	0	R	**SEGD2H Register Busy**
	Set when the value written to SEGD2H is being synchronized.			
9	SEGD1H	0	R	**SEGD1H Register Busy**
	Set when the value written to SEGD1H is being synchronized.			
8	SEGD0H	0	R	**SEGD0H Register Busy**
	Set when the value written to SEGD0H is being synchronized.			
7	SEGD3L	0	R	**SEGD3L Register Busy**
	Set when the value written to SEGD3L is being synchronized.			
6	SEGD2L	0	R	**SEGD2L Register Busy**
	Set when the value written to SEGD2L is being synchronized.			

Bit	Name	Reset	Access	Description
5	SEGD1L	0	R	**SEGD1L Register Busy**
	Set when the value written to SEGD1L is being synchronized.			
4	SEGD0L	0	R	**SEGD0L Register Busy**
	Set when the value written to SEGD0L is being synchronized.			
3	AREGB	0	R	**AREGB Register Busy**
	Set when the value written to AREGB is being synchronized.			
2	AREGA	0	R	**AREGA Register Busy**
	Set when the value written to AREGA is being synchronized.			
1	BACTRL	0	R	**BACTRL Register Busy**
	Set when the value written to BACTRL is being synchronized.			
0	CTRL	0	R	**CTRL Register Busy**
	Set when the value written to CTRL is being synchronized.			

16.5.31 LCD_FRAMERATE - Frame Rate

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																												0x000				
Access																												RW				
Name																												FRDIV				

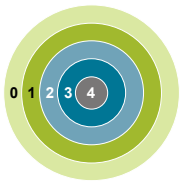
Bit	Name	Reset	Access	Description
31:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	FRDIV	0x000	RW	**Frame Rate Divider**
	Determines number of prescaled clocks per phase. Static has 2 phases, and octaplex has sixteen phases per frame.			

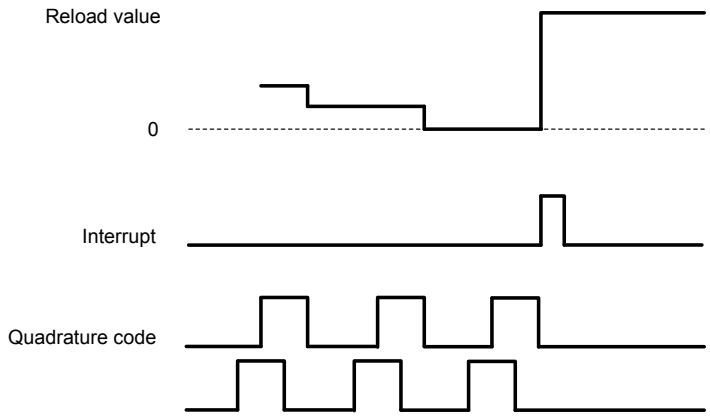
16.5.32 LCD_SEGEN2 - Segment Enable (32 to 39)

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																									0x00							
Access																									RW							
Name																									SEGEN2							

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	SEGEN2	0x00	RW	**Segment Enable (second Group)** Determines which segment lines are enabled from (32 to 39). The GPIO pin also needs to be configured as DISABLED in the GPIO pin configuration.

17. PCNT - Pulse Counter





Quick Facts

What?

The Pulse Counter (PCNT) decodes incoming pulses. The module has a quadrature mode which may be used to decode the speed and direction of a mechanical shaft. PCNT can operate in EM0 Active down to EM3 Stop.

Why?

The PCNT generates an interrupt after a specific number of pulses (or rotations), eliminating the need for timing or I/O interrupts and CPU processing to measure pulse widths, etc.

How?

PCNT uses the LFACLK or may be externally clocked from a pin. The module incorporates a 16-bit up/down-counter to keep track of incoming pulses or rotations.

17.1 Introduction

The Pulse Counter (PCNT) can be used for counting incoming pulses on a single input or to decode quadrature encoded inputs in EM0 Active down to EM3 Stop. It can run from the internal LFACLK while counting pulses on the PCNTn_S0IN pin. Or, alternately, the PCNTn_S0IN pin may be used as an external clock source that runs both the PCNT counter and register access.

17.2 Features

- 16-bit counter with reload register
- Auxiliary counter for counting a single direction
- Single input oversampling up/down counter mode
- Externally clocked single input pulse up/down counter mode
- Quadrature decoder modes
  - Externally clocked quadrature decoder 1X mode
  - Oversampling quadrature decoder 1X, 2X and 4X modes
- Interrupt on counter underflow and overflow
- Interrupt when a direction change is detected (quadrature decoder mode only)
- Optional pulse width filter
- Optional input inversion/edge detect select
- Optional inputs from PRS
- Asynchronously triggered compare and clear

17.3 Functional Description

An overview of the PCNT module is shown in Figure 17.1 PCNT Overview on page 626.

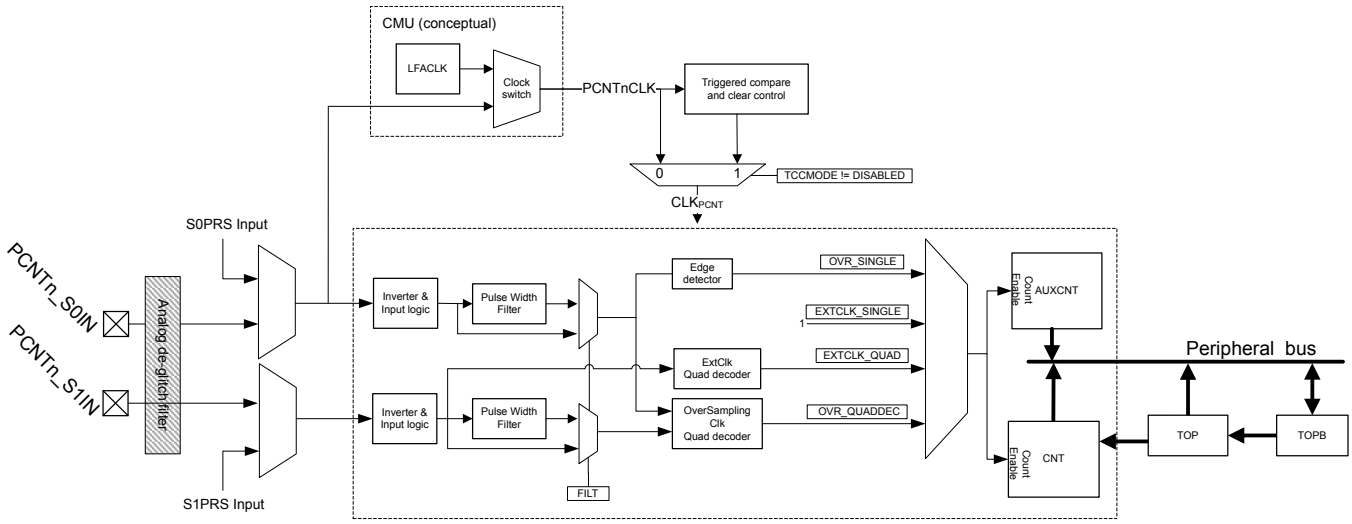


Figure 17.1. PCNT Overview

17.3.1 Pulse Counter Modes

The pulse counter can operate in single input oversampling mode (OVSSINGLE), externally clocked single input counter mode (EXTCLKSINGLE), externally clocked quadrature decoder mode (EXTCLKQUAD) and oversampling quadrature decoder modes(OVSQUAD1X, OVSQUAD2X and OVSQUAD4X). The following sections describe operation of each of these modes and how they are enabled. Input timing constraints are described in 17.3.6 Clock Sources and 17.3.7 Input Filter.

17.3.1.1 Single Input Oversampling Mode

This mode is enabled by writing OVSSINGLE to the MODE field in the PCNTn_CTRL register and disabled by writing DISABLE to the same field. The LFACLK clock source to the pulse counter is configured by clearing PCNT0CLKSEL in the CMU_PCNTCTRL in the Clock Management Unit (CMU).

The optional pulse width filter is enabled by setting the FILT bit in the PCNTn_CTRL register. Additionally, the PCNTn_S0IN input may be inverted, so that falling edges are counted, by setting the EDGE bit in the PCNTn_CTRL register.

If S1CDIR in the PCNTn_CTRL register is cleared, PCNTn_S0IN is the only observed input in this mode. The PCNTn_S0IN input is sampled by the LFACLK and the number of detected positive or negative edges on PCNTn_S0IN appears in PCNTn_CNT. The counter may be configured to count down by setting the CNTDIR bit in PCNTn_CTRL. Default is to count up.

The counting direction can also be controlled externally in this mode by setting S1CDIR. This will make the input value on PCNTn_S1IN decide the direction counted on a PCNTn_S0IN edge. If PCNTn_S1IN is high, the count is done according to CNTDIR in PCNTn_CTRL. If low, the count direction is opposite.

17.3.1.2 Externally Clocked Single Input Counter Mode

This mode is enabled by writing EXTCLKSINGLE to the MODE field in the PCNTn_CTRL register and disabled by writing DISABLE to the same field. The external pin clock source is configured by setting PCNT0CLKSEL in the CMU_PCNTCTRL register (10. CMU - Clock Management Unit).

Positive edges on PCNTn_S0IN are used to clock the counter. Similar to the oversampled mode, PCNTn_S1IN is used to determine the count direction if S1CDIR is set. If not, CNTDIR in PCNTn_CTRL solely defines count direction.

The digital pulse width filter is not available in this mode. The analog de-glitch filter in the GPIO pads is capable of removing some unwanted noise. However, this mode may be susceptible to spikes and unintended pulses from devices such as mechanical switches, and is therefore most suited to take input from electronic sensors etc. that generate single wire pulses.

17.3.1.3 Quadrature Decoder Modes

Two different types of quadrature decoding is supported in the pulse counter: the externally clocked (Asynchronous) quadrature decoding and the oversampling (Synchronous) quadrature decoding. The externally clocked mode supports 1X quadrature decoding whereas the oversampling mode supports 1X, 2X and 4X quadrature decoding. These modes are described in detail in 17.3.1.4 Externally Clocked Quadrature Decoder Mode and 17.3.1.5 Oversampling Quadrature Decoder Mode .

17.3.1.4 Externally Clocked Quadrature Decoder Mode

This mode is enabled by writing EXTCLKQUAD to the MODE field in PCNTn_CTRL and disabled by writing DISABLE to the same field. The external pin clock source is configured by setting PCNT0CLKSEL in the CMU_PCNTCTRL register (10. CMU - Clock Management Unit).

In this mode, both edges on PCNTn_S0IN pin are used to sample PCNTn_S1IN pin, in order to decode the quadrature code. A quadrature coded signal contains information about the relative speed and direction of a rotating shaft as illustrated by Figure 17.2 PCNT Quadrature Coding on page 628, hence the direction of the counter register PCNTn_CNT is controlled automatically.

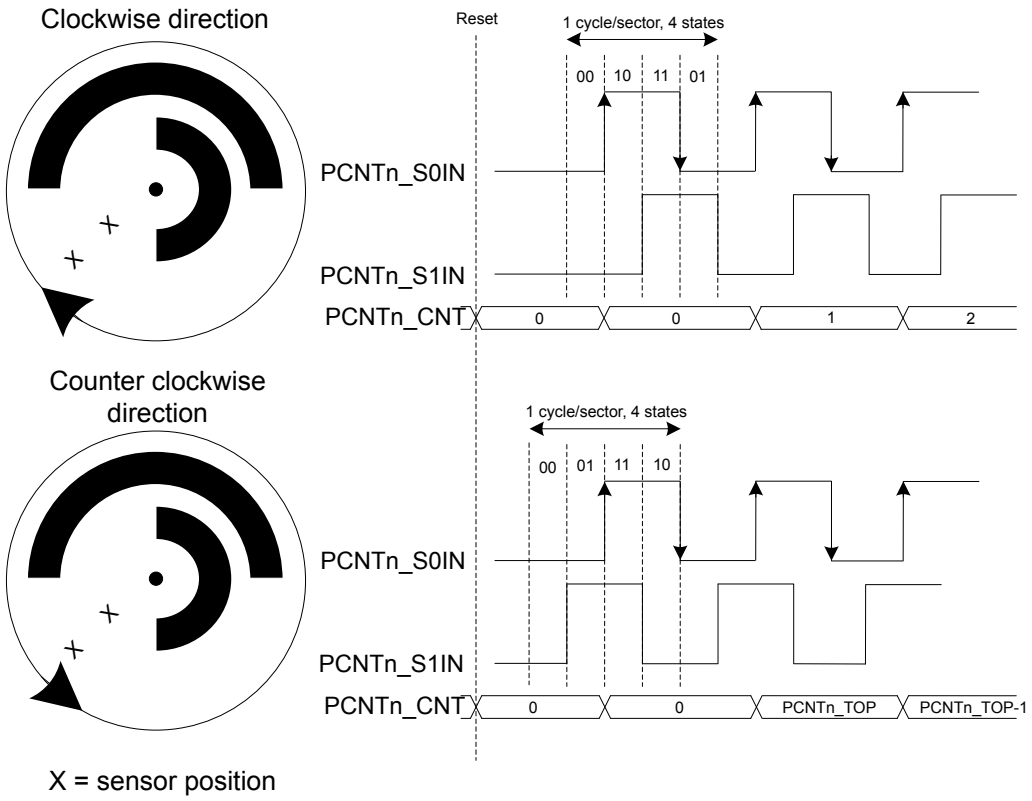


Figure 17.2. PCNT Quadrature Coding

If PCNTn_S0IN leads PCNTn_S1IN in phase, the direction is clockwise, and if it lags in phase the direction is counter-clockwise. Default behavior is illustrated by Figure 17.2 PCNT Quadrature Coding on page 628.

The counter direction may be read from the DIR bit in the PCNTn_STATUS register. Additionally, the DIRCNG interrupt in the PCNTn_IF register is generated when a direction change is detected. When a change is detected, the DIR bit in the PCNTn_STATUS register must be read to determine the current new direction.

Note: The sector disc illustrated in the figure may be finer grained in some systems. Typically, they may generate 2-4 PCNTn_S0IN wave periods per 360° rotation.

The direction of the quadrature code and control of the counter is generated by the simple binary function outlined by Table 17.1 PCNT QUAD Mode Counter Control Function on page 628. Note that this function also filters some invalid inputs that may occur when the shaft changes direction or temporarily toggles direction.

Table 17.1. PCNT QUAD Mode Counter Control Function

Inputs		Control/Status	
S1IN posedge	S1IN negedge	Count Enable	CNTDIR status bit
0	0	0	0

Inputs		Control/Status	
S1IN posedge	S1IN negedge	Count Enable	CNTDIR status bit
0	1	1	0
1	0	1	1
1	1	0	0
Note: PCNTn_S1IN is sampled on both edges of PCNTn_S0IN.			

17.3.1.5 Oversampling Quadrature Decoder Mode

There are three Oversampling Quadrature Decoder Modes supported: 1X , 2X and 4X. These modes are enabled by writing OVSQUAD1X, OVSQUAD2X and OVSQUAD4X, respectively, to the MODE field in PCNTn_CTRL and disabled by writing DISABLE to the same field. The LFACLK clock source to the pulse counter must be configured by clearing PCNT0CLKSEL in the CMU_PCNTCTRL in the Clock Management Unit (CMU), 10. CMU - Clock Management Unit .

The optional pulse width filter is enabled by setting the FILT bit in the PCNTn_CTRL register. The filter applies to both inputs PCNTn_S0IN and PCNTn_S1IN. The filter length is configured by FILTLEN in PCNTn_OVSCFG register.

Based on the modes selected, the decoder updates the counter on different events. In the OVSQUAD1X mode, the counter is updated on the rising edge of the PCNTn_S0IN input when counting up, and on the negedge of the PCNTn_S0IN input when counting down. In the OVSQUAD2X mode, the counter is updated on both edges of PCNTn_S0IN input. In the OVSQUAD4X mode the counter is updated on both edges of both inputs PCNTn_S0IN and PCNTn_S1IN. Table 17.2 PCNT OVSQUAD 1X, 2X and 4X Mode Counter Control Function on page 630 outlines the increment or decrement of the counter based on the Quadrature Mode selected.

Note: The decoding behavior of OVSQUAD1X mode is slightly different compared to EXTCLKQUAD mode(also 1X mode). In the EXTCLKQUAD mode, the counter is updated only on the posedge of S0IN input. However, in the OVSQUAD1X mode, the counter is updated on the posedge of S0IN when counting up and on the negedge of S0IN when counting down.

Table 17.2. PCNT OVSQUAD 1X, 2X and 4X Mode Counter Control Function

Direction	Previous State		Next State		OVSQUAD MODE		
	S1IN	S0IN	S1IN	S0IN	1X	2X	4X
Clockwise	0	0	0	1	+1	+1	+1
	0	1	1	1			+1
	1	1	1	0		+1	+1
	1	0	0	0			+1
Counter Clockwise	1	0	1	1		-1	-1
	1	1	0	1			-1
	0	1	0	0	-1	-1	-1
	0	0	1	0			-1

Figure 17.3 PCNT State Transitions for Different Oversampling Quadrature Decoder Modes on page 631 illustrates the different states of the quadrature input and the state transitions that updates the counter for the different modes. Each cycle of the input states results in 1 update, 2 updates and 4 updates of the counter for OVSQUAD1X, OVSQUAD2X and OVSQUAD4X modes respectively.

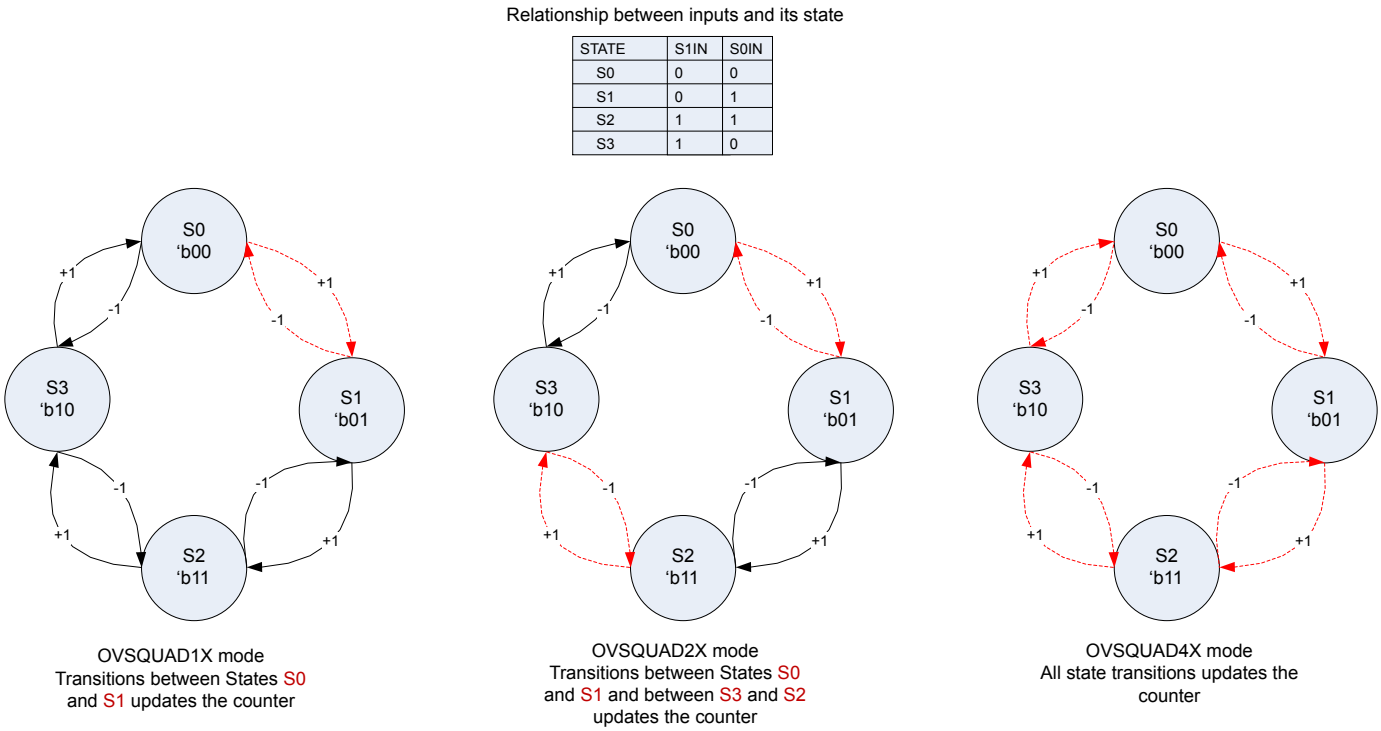


Figure 17.3. PCNT State Transitions for Different Oversampling Quadrature Decoder Modes

The counter direction can be read from the DIR bit in PCNTn_STATUS register. Additionally, the DIRCNG interrupt in the PCNTn_IF is generated when the direction change is detected. When a change is detected, the DIR bit in the PCNTn_STATUS register must be read to determine the new direction.

In the oversampling quadrature decoder modes, the maximum input toggle frequency supported is 8KHz. For frequencies of 8KHz and higher, incorrect decoding occurs. The different decoding modes and the counter updates are further illustrated by Figure 17.4 PCNT Oversampling Quadrature Decoder 1X Mode on page 631, Figure 17.5 PCNT Oversampling Quadrature Decoder 2X Mode on page 632 and Figure 17.6 PCNT Oversampling Quadrature Decoder 4X Mode on page 632.

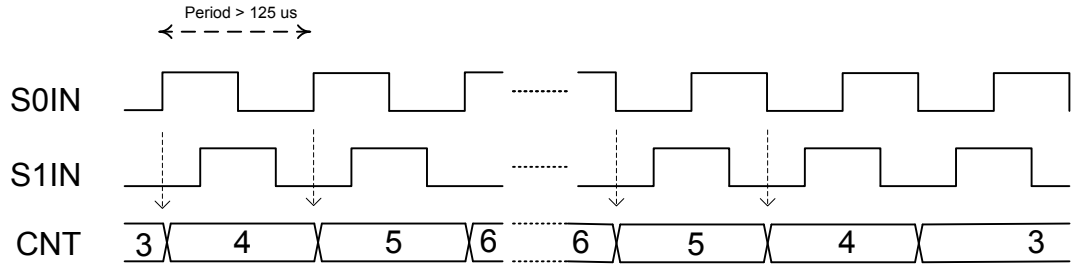


Figure 17.4. PCNT Oversampling Quadrature Decoder 1X Mode

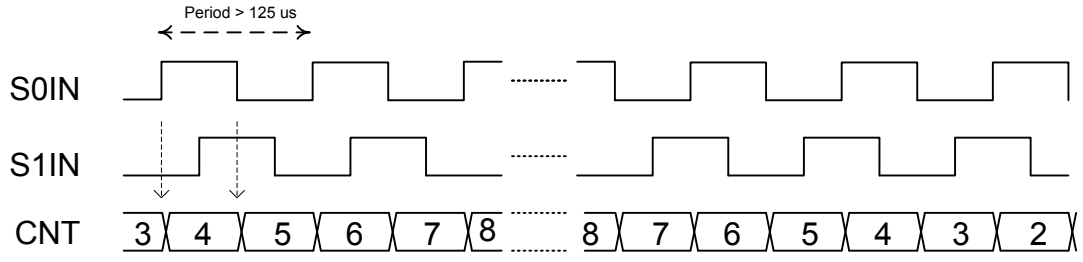


Figure 17.5. PCNT Oversampling Quadrature Decoder 2X Mode

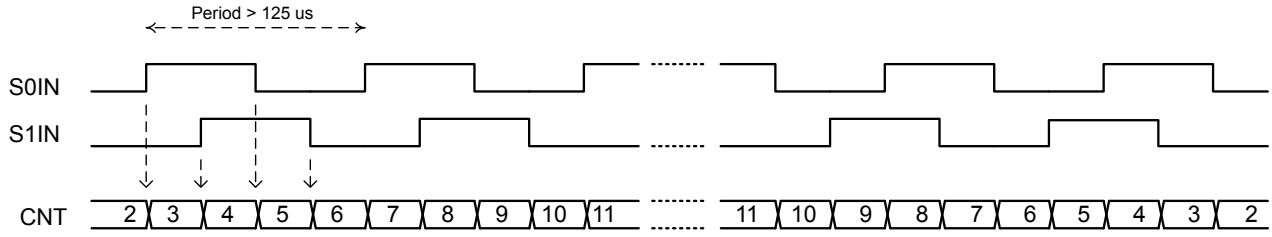


Figure 17.6. PCNT Oversampling Quadrature Decoder 4X Mode

The above modes, by default are prone to flutter effects in the inputs PCNTn_S0IN and PCNTn_S1IN. When this occurs, the counter changes directions rapidly causing DIRCNG interrupts and unnecessarily waking the core. To prevent this, set FLUTTERRM in PCNTn_OVSCFG register. When enabled, flutter is removed, thus preventing unnecessary wakeup of the core. The flutter removal logic works by preventing update of the counter value if the wheel keeps changing direction as a result of flutter. The counter is only updated if the current and previous state transition of the rotation are in the same direction. These state transitions are quadrature decoder mode specific. The highlighted state transitions in Figure 17.3 PCNT State Transitions for Different Oversampling Quadrature Decoder Modes on page 631 are the ones considered for the different quadrature decoder modes. Figure 17.7 PCNT Oversampling Quadrature Decoder with Flutter Removal on page 632 shows how the counter is updated for the different quadrature decoder modes with flutter removal FLUTTERRM enabled in PCNTn_OVSCFG.

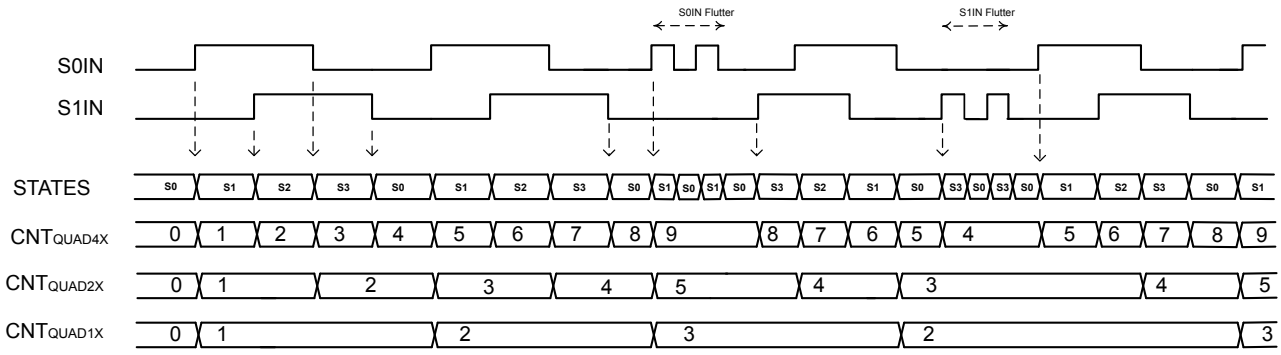


Figure 17.7. PCNT Oversampling Quadrature Decoder with Flutter Removal

17.3.2 Hysteresis

By default the pulse counter wraps to 0 when passing the configured top value, and wraps to the top value when counting down from 0. On these events, a system will likely want to wake up to store and track the overflow count. This is fine if the pulse counter is tracking a monotonic value or a value that does not change directions frequently. In the latter scenario, if the counter changes directions around the overflow/underflow point, the system will have to wake up frequently to keep track of the rotations, resulting in higher current consumption.

To solve this, the pulse counter has a way of introducing hysteresis to the counter. When HYST in PCNTn_CTRL is set, the pulse counter will always wrap to TOP/2 on underflows and overflows. This takes the counter away from the area where it might overflow or underflow, removing the problem. Figure 17.8 PCNT Hysteresis behavior of Counter on page 633 illustrates the hysteresis behavior.

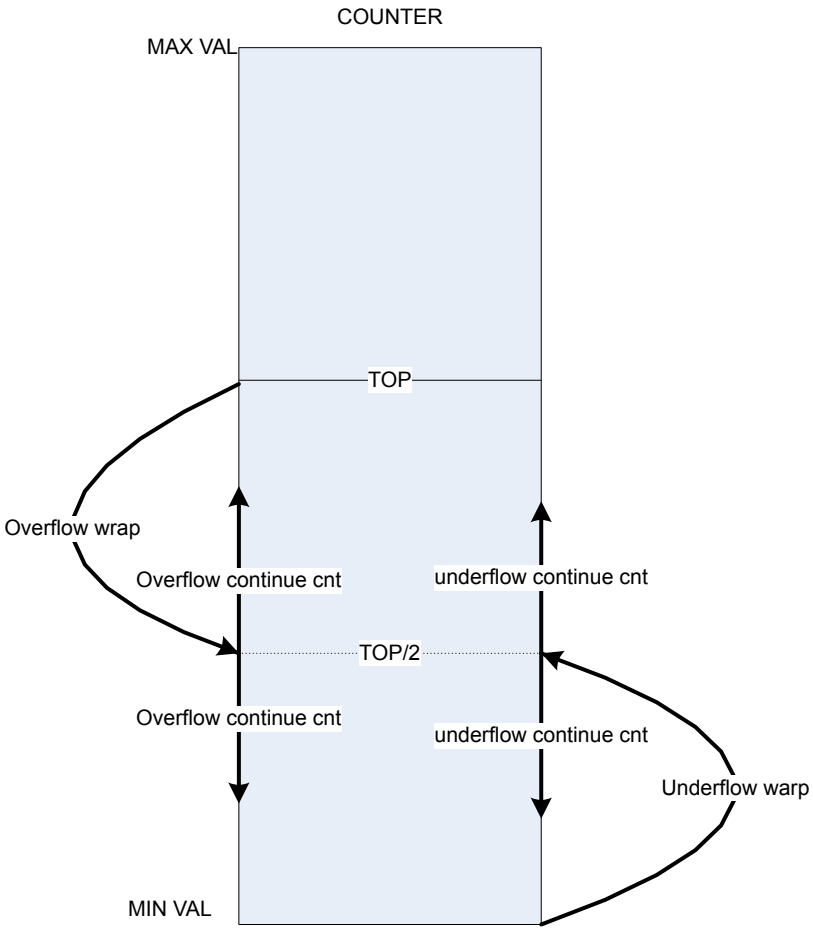


Figure 17.8. PCNT Hysteresis behavior of Counter

Given a starting value of 0 for the counter, the absolute count value when hysteresis is enabled can be calculated with the equations Figure 17.9 Absolute Position With Hysteresis and Even TOP Value on page 633 or Figure 17.10 Absolute Position With Hysteresis and Odd TOP Value on page 633, depending on whether the TOP value is even or odd.

$$CNT_{abs} = CNT - UF_{CNT} \times (TOP/2+1) + OF_{CNT} \times (TOP/2+1)$$

Figure 17.9. Absolute Position With Hysteresis and Even TOP Value

$$CNT_{abs} = CNT - UF_{CNT} \times (TOP/2+1) + OF_{CNT} \times (TOP/2+2)$$

Figure 17.10. Absolute Position With Hysteresis and Odd TOP Value

17.3.3 Auxiliary Counter

To be able to keep explicit track of counting in one direction in addition to the regular counter which counts both up and down, the auxiliary counter can be used. The pulse counter can, for instance, be configured to keep track of the absolute rotation of the wheel, while at the same time the auxiliary counter can keep track of how much the wheel has reversed.

The auxiliary counter is enabled by configuring AUXCNTEV in PCNTn_CTRL. It will always count up, but it can be configured whether it should count up on up-events, down-events or both, keeping track of rotation either way or general movement. The value of the auxiliary counter can be read from the PCNTn_AUXCNT register.

Overflows on the auxiliary counter happen when the auxiliary counter passes the top value of the pulse counter, configured in PCNTn_TOP. In that event, the AUXOF interrupt flag is set, and the auxiliary counter wraps to 0.

As the auxiliary counter, the main counter can be configured to count only on certain events. This is done through CNTEV in PCNTn_CTRL, and it is possible like for the auxiliary counter, to make the main counter count on only up and down events. The difference between the counters is that where the auxiliary counter will only count up, the main counter will count up or down depending on the direction of the count event.

17.3.4 Triggered Compare and Clear

The pulse counter features triggered compare and clear. When enabled, a configurable trigger will induce a comparison between the main counter, PCNTn_CNT, and the top value, PCNTn_TOP. After the comparison, the counter is cleared. The trigger for a compare and clear event is configured in the TCCMODE bit-field in PCNTn_CTRL. There are two options, LFA and PRS. If LFA is selected, the pulse counter will be compared with the top value, and cleared every 2^N LFA clock cycle (where N is the value of TCCPRESC in PCNTn_CTRL). If a PRS trigger is selected, the active PRS channel is configured in TCCPRSSEL in PCNTn_CTRL. The PRS input can be inverted by setting TCCPRSPOL, triggering the compare and clear on the negative edge of the PRS input. The PRS input can also be used as a gate for the pulse counter clock. This is enabled by setting PRSGATEEN in PCNTn_CTRL.

Note: When PRSGATEEN is set, the clock to the entire pulse counter will be gated by the PRS input, meaning that register writes will not take effect while the gated clock is inactive.

Comparison with PCNTn_TOP can be performed in three ways: range, greater than or equal, and less than or equal. TCCCOMP in PCNTn_CTRL configures comparison mode. Upon a compare match, the TCC interrupt is set, and the PRS output from the pulse counter is set. The PRS output will remain set until the next compare and clear event. Triggered compare and clear is intended for use when the pulse counter is configured to count up. In this mode, PCNTn_CNT will not wrap to 0 when hitting PCNTn_TOP, it will keep counting. In addition, the counter will not overflow, it will rather stop counting, just setting the overflow interrupt flag.

Figure 17.11 PCNT Triggered Compare and Clear on page 635 shows an overview of the control circuitry for triggered compare and clear. The control circuitry includes two positive edge detectors (PED) and glitch filters, used to generate clocks for the pulse counter. The two clock outputs are mutually exclusive: If both edge detectors receive a pulse at the same time, the output pulse from one of them will be postponed until the other edge detectors output pulse has completed.

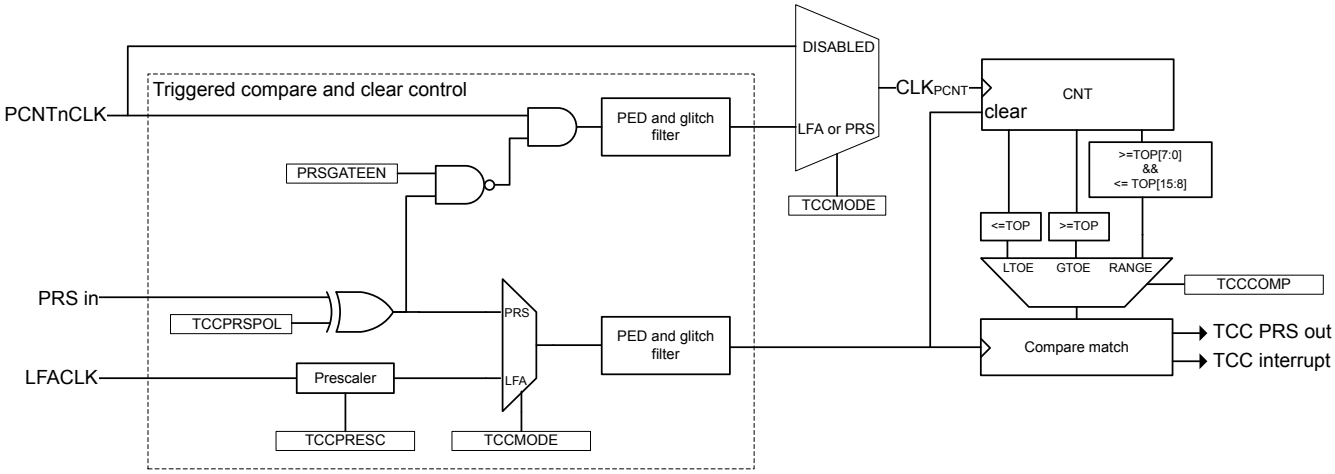


Figure 17.11. PCNT Triggered Compare and Clear

Note: TCCMODE, TCCPRESC, PRSGATEEN, TCCPRSPOL, and TCCPRSSEL in PCNTn_CTRL should only be altered when RSTEN in PCNTn_CTRL is set.

17.3.5 Register Access

The counter-clock domain may be clocked externally. To update the counter-clock domain registers from software in this mode, 2-3 clock pulses on the external clock are needed to synchronize accesses to the externally clocked domain. Clock source switching is controlled from the registers in the CMU (10. CMU - Clock Management Unit).

When the RSTEN bit in the PCNTn_CTRL register is set, the PCNT clock domain is asynchronously held in reset. The reset is synchronously released two PCNT clock edges after the RSTEN bit in the PCNTn_CTRL register is cleared by software. This asynchronous reset restores the reset values in PCNTn_TOP, PCNTn_CNT and other control registers in the PCNT clock domain.

CNTRSTEN works in a similar manner as RSTEN, but only resetting the counter, CNT. Note that the counter is also reset by RSTEN.

AUXCNTRSTEN works in a similar manner as RSTEN, but only resetting the auxiliary counter, PCNTn_AUXCNT. Note that the auxiliary counter is also reset by RSTEN.

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. Refer to 4.3 Access to Low Energy Peripherals (Asynchronous Registers) for a description on how to perform register accesses to Low Energy Peripherals.

Note: PCNTn_TOP and PCNTn_CNT are read-only registers. When writing to PCNTn_TOPB, make sure that the counter value, PCNTn_CNT, can not exceed the value written to PCNTn_TOPB within two clock cycles.

17.3.6 Clock Sources

The pulse counter may be clocked from two possible clock sources: LFACLK or an external clock. The clock selection is configured by the PCNT0CLKSEL bit in the CMU_PCNTCTRL in the Clock Management Unit (CMU), 10. CMU - Clock Management Unit . The default clock source is the LFACLK.

This PCNT module may also use PCNTn_S0IN as an external clock to clock the counter (EXTCLKSINGLE mode) and to sample PCNTn_S1IN (EXTCLKQUAD mode). Setup, hold and max frequency constraints for PCNTn_S0IN and PCNTn_S1IN for these modes are specified in the device data sheet.

To use this module, the LE interface clock must be enabled in CMU_HFBUSCLKEN0, in addition to the module clock in CMU_PCNTCTRL.

Note: PCNT Clock Domain Reset, RSTEN, should be set when changing clock source for PCNT. If changing to an external clock source, the clock pin has to be enabled as input prior to de-asserting RSTEN. Changing clock source without asserting RSTEN results in undefined behaviour.

17.3.7 Input Filter

An optional pulse width filter is available in OVSSINGLE and OVSQUAD modes, when LFACLK is selected as a clock source for the Pulse Counter in CMU 10. CMU - Clock Management Unit . The filter is enabled by writing 1 to the FILT bit in the PCNTn_CTRL register. When enabled, the high and low periods of PCNTn_S0IN and PCNTn_S1IN must be stable for a programmable number of consecutive clock cycles before the edge is passed to the edge detector. The filter length should be programmed in FILTLEN field of the PCNTn_OVSCFG register.

The filter length is given by Figure 17.12 PCNT Input Filter Length Equation on page 636:

Filter length = (FILTLEN + 5) LFACLK cycles

Figure 17.12. PCNT Input Filter Length Equation

The maximum filter length configured is 260 LFACLK cycles.

In EXTCLKSINGLE and EXTCLKQUAD mode, there is no digital pulse width filter available.

17.3.8 Edge Polarity

The edge polarity can be set by configuring the EDGE bit in the PCNTn_CTRL register. When this bit is cleared, the pulse counter counts positive edges of PCNTn_S0IN input. When this bit is set, the pulse counter counts negative edges in OVSSINGLE mode. Also, when the EDGE bit is set in the OVSSINGLE and EXTCLKSINGLE modes, the PCNTn_S1IN input is inverted. In OVSQUAD 1X-4X modes the EDGE bit inverts both inputs.

Note: The EDGE bit in PCNTn_CTRL has no effect in EXTCLKQUAD mode.

17.3.9 PRS and PCNTn_S0IN,PCNTn_S1IN Inputs

It is possible to receive input from PRS on both PCNTn_S0IN (or PCNTn_S1IN) by setting S0PRSEN (or S1PRSEN) in PCNTn_INPUT. The PRS channel used can be selected using S0PRSSEL (or S1PRSSEL) in PCNTn_INPUT.

In the Oversampling quadrature decoder modes, the input frequency should be less than 8KHz to ensure correct functionality.

PCNT module generates three PRS outputs the TCC PRS output, the CNT OF/UF PRS output and the CNT DIR PRS output. The TCC PRS is generated on compare match of TCC event. The CNT OF/UF combined PRS is generated when the counter overflow or underflows. The CNT DIR PRS is a level PRS and indicates the current direction of count of counter CNT

Note: S0PRSEN,S1PRSEN,S0PRSSEL,S1PRSSEL should only be altered when RSTEN in PCNTn_CTRL is set.

17.3.10 Interrupts

The interrupt generated by PCNT uses the PCNTn_INT interrupt vector. Software must read the PCNTn_IF register to determine which module interrupt that generated the vector invocation.

17.3.10.1 Underflow and Overflow Interrupts

The underflow interrupt flag (UF) is set when the counter counts down from 0. I.e. when the value of the counter is 0 and a new pulse is received. The PCNTn_CNT register is loaded with the PCNTn_TOP value after this event.

The overflow interrupt flag (OF) is set when the counter counts up from the PCNTn_TOP (reload) value. I.e. if PCNTn_CNT = PCNTn_TOP and a new pulse is received. The PCNTn_CNT register is loaded with the value 0 after this event.

17.3.10.2 Direction Change Interrupt

The PCNTn_PCNT module sets the DIRCNG interrupt flag (PCNTn_IF register) for EXTCLKQUAD and OVSQUAD1X-4X modes when the direction of the quadrature code changes. The behavior of this interrupt in the EXTCLKQUAD mode is illustrated by Figure 17.13 PCNT Direction Change Interrupt (DIRCNG) Generation on page 638.

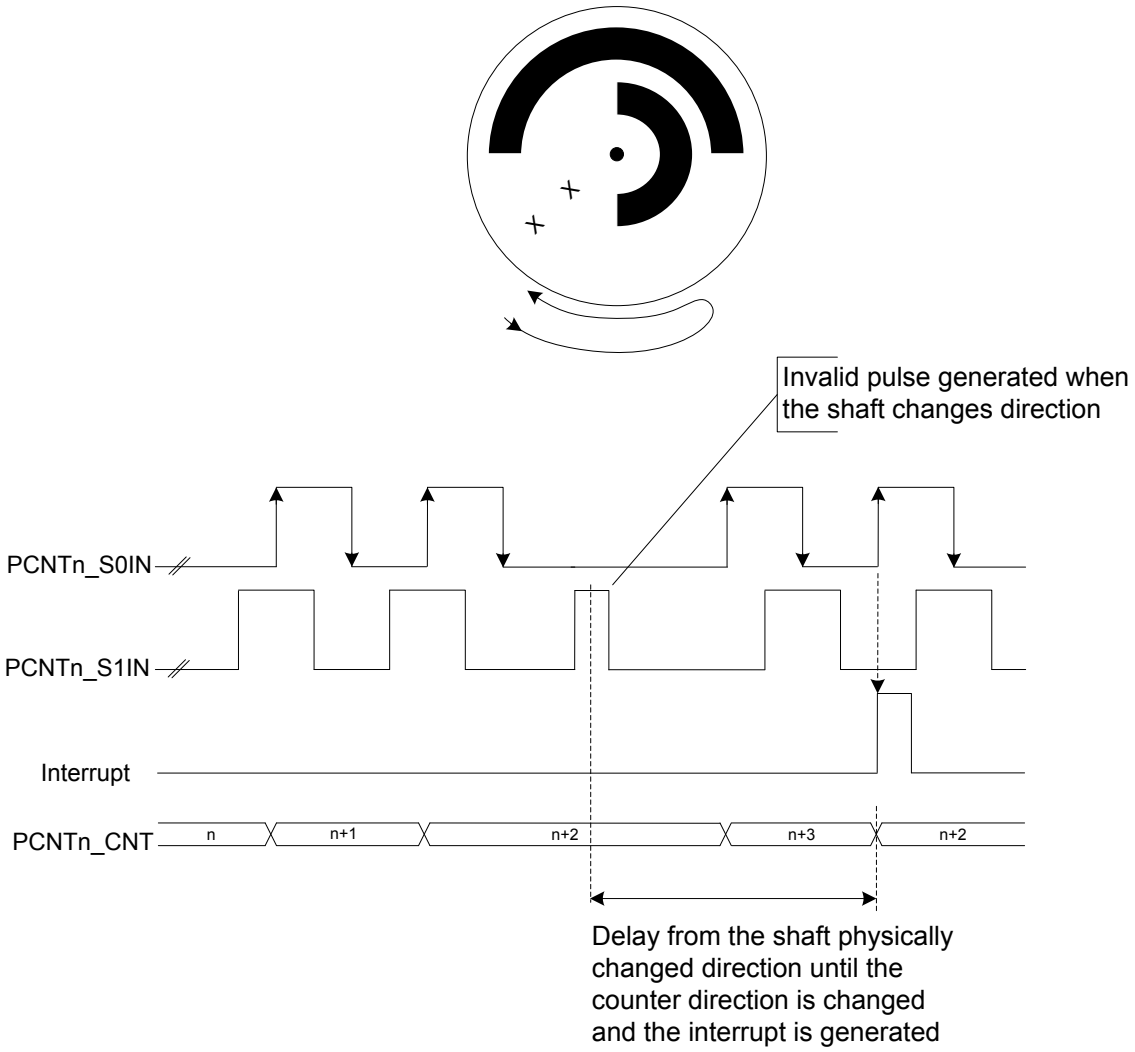


Figure 17.13. PCNT Direction Change Interrupt (DIRCNG) Generation

17.3.11 Cascading Pulse Counters

When two or more Pulse Counters are available, it is possible to cascade them. For example two 16-bit Pulse Counters can be cascaded to form a 32-bit pulse counter. This can be done with the help of the CNT UF/OF PRS and CNT DIR PRS ouputs. The figure Figure 17.14 PCNT Cascading to two 16-bit PCNT to form a 32-bit PCNT on page 639 illustrates this structure.

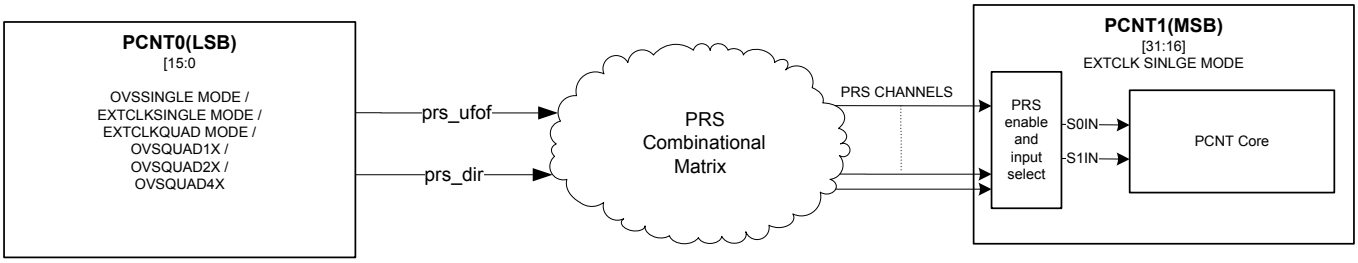


Figure 17.14. PCNT Cascading to two 16-bit PCNT to form a 32-bit PCNT

For cascading of Pulse Counters to work, the PCNT1 according to the figure Figure 17.14 PCNT Cascading to two 16-bit PCNT to form a 32-bit PCNT on page 639 should be programmed in EXTCLKSINGLE mode and its S0IN and S1IN inputs should be configured to prs_ufof and prs_dir of PCNT0 respectively. In addition to this, a strict programming sequence needs to be followed to ensure both PCNTs are in sync with each other.

- Configure PCNT0 registers. eg. PCNT0_INPUT,PCNT0_CTRL,PCNT0_OVSCFG etc.
- Wait for PCNT0_SYCNBUSY to be cleared to ensure the registers are synchronized to the asynchronous clock domain.
- Hold PCNT0 in sw reset by setting PCNT0_CTRL_RSTEN.
- Configure PCNT1_CTRL to EXTCLKSINLE mode with S1CDIR and CNTDIR bit set. Configure INPUT to accept "prs_ufof" and "prs_dir" of PCNT0 on S0IN and S1IN respectively.
- Wait for PCNTn_SYCNBUSY to be cleared to ensure the registers are synchronized to the asynchronous clock domain. Use three PRS_SWPULSE on the S0IN prs channel to ensure this synchronization.
- Hold PCNT1 in sw reset by setting PCNT1_CTRL_RSTEN.
- Clear PCNT1_CTRL_RSTEN and synchronize it by asserting two PRS_SWPULSE on the S0IN input.
- Finally clear PCNT0_CTRL_RSTEN and start counting.

Note: When RSTEN in PCNTn_CTRL is set, the TOP value in the Pulse Counter gets cleared. Therefore, in order to update the TOP value while RSTEN is set, assert TOPBHFEN bit in PCNTn_CTRL. This will update the TOP value with the TOPB value even without having to synchronize the TOPB value. This only works if TOPBHFEN and TOPB are configured while RSTEN in PCNTn_CTRL is set.

17.4 Register Map

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

Offset	Name	Type	Description
0x000	PCNTn_CTRL	RW	Control Register
0x004	PCNTn_CMD	W1	Command Register
0x008	PCNTn_STATUS	R	Status Register
0x00C	PCNTn_CNT	R	Counter Value Register
0x010	PCNTn_TOP	R	Top Value Register
0x014	PCNTn_TOPB	RW	Top Value Buffer Register
0x018	PCNTn_IF	R	Interrupt Flag Register
0x01C	PCNTn_IFS	W1	Interrupt Flag Set Register
0x020	PCNTn_IFC	(R)W1	Interrupt Flag Clear Register
0x024	PCNTn_IEN	RW	Interrupt Enable Register
0x02C	PCNTn_ROUTELOC0	RW	I/O Routing Location Register
0x040	PCNTn_FREEZE	RW	Freeze Register
0x044	PCNTn_SYNCBUSY	R	Synchronization Busy Register
0x064	PCNTn_AUXCNT	R	Auxiliary Counter Value Register
0x068	PCNTn_INPUT	RW	PCNT Input Register
0x06C	PCNTn_OVSCFG	RW	Oversampling Config Register

17.5 Register Description

17.5.1 PCNTn_CTRL - Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0		0x0				0	0	0x0			0x0			0x0		0	0	0x0		0x0		0	0	0	0	0	0	0	0x0		
Access	RW		RW				RW	RW	RW			RW			RW		RW	RW	RW		RW		RW	RW	RW	RW	RW	RW	RW	RW		
Name	TOPBHFSEL		TCCPRSSEL				TCCPRSPOL	PRSGATEEN	TCCCOMP			TCCPRESC			TCCMODE		EDGE	CNTDIR	AUXCNTEV		CNTEV		S1CDIR	HYST	DEBUGHALT	AUXCNTRSTEN	CNTRSTEN	RSTEN	FILT	MODE		

Bit	Name	Reset	Access	Description
31	TOPBHFSEL	0	RW	**TOPB High Frequency Value Select**
	Apply High frequency value of TOPB to TOP register. Should be used only when RSTEN in PCNTn_CTRL is set			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29:26	TCCPRSSEL	0x0	RW	**TCC PRS Channel Select**
	Select PRS channel used as compare and clear trigger.			

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected.
1	PRSCH1	PRS Channel 1 selected.
2	PRSCH2	PRS Channel 2 selected.
3	PRSCH3	PRS Channel 3 selected.
4	PRSCH4	PRS Channel 4 selected.
5	PRSCH5	PRS Channel 5 selected.
6	PRSCH6	PRS Channel 6 selected.
7	PRSCH7	PRS Channel 7 selected.
8	PRSCH8	PRS Channel 8 selected.
9	PRSCH9	PRS Channel 9 selected.
10	PRSCH10	PRS Channel 10 selected.
11	PRSCH11	PRS Channel 11 selected.
12	PRSCH12	PRS Channel 12 selected.
13	PRSCH13	PRS Channel 13 selected.
14	PRSCH14	PRS Channel 14 selected.
15	PRSCH15	PRS Channel 15 selected.

Bit	Name	Reset	Access	Description
25	TCCPRSPOL	0	RW	**TCC PRS Polarity Select**

Configure which edge on the PRS input is used to trigger a compare and clear event

Value	Mode	Description
0	RISING	Rising edge on PRS trigger compare and clear event.
1	FALLING	Falling edge on PRS trigger compare and clear event.

Bit	Name	Reset	Access	Description
24	PRSGATEEN	0	RW	**PRS Gate Enable**

When set, the clock input to the pulse counter will be gated when the selected PRS input is the inverse of TCCPRSPOL.

Bit	Name	Reset	Access	Description
23:22	TCCCOMP	0x0	RW	**Triggered Compare and Clear Compare Mode**

Selects the mode for comparison upon a compare and clear event.

Value	Mode	Description
0	LTOE	Compare match if PCNT_CNT is less than, or equal to PCNT_TOP.
1	GTOE	Compare match if PCNT_CNT is greater than or equal to PCNT_TOP.
2	RANGE	Compare match if PCNT_CNT is less than, or equal to PCNT_TOP[15:8]], and greater than, or equal to PCNT_TOP[7:0].

Bit	Name	Reset	Access	Description
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
20:19	TCCPRESC	0x0	RW	**Set the LFA Prescaler for Triggered Compare and Clear**

Selects the prescaler value for LFA compare and clear events

Value	Mode	Description
0	DIV1	Compare and clear event each LFA cycle.
1	DIV2	Compare and clear performed on every other LFA cycle.
2	DIV4	Compare and clear performed on every 4th LFA cycle.
3	DIV8	Compare and clear performed on every 8th LFA cycle.

Bit	Name	Reset	Access	Description
18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
17:16	TCCMODE	0x0	RW	**Sets the Mode for Triggered Compare and Clear**

Selects whether compare and clear should be triggered on each LFA clock, or from PRS

Value	Mode	Description
0	DISABLED	Triggered compare and clear not enabled.
1	LFA	Compare and clear performed on each (optionally prescaled) LFA clock cycle.
2	PRS	Compare and clear performed on positive PRS edges.

Bit	Name	Reset	Access	Description
15	EDGE	0	RW	**Edge Select**

Determines the polarity of the incoming edges. This bit should be written when PCNT is in DISABLE mode, otherwise the behavior is unpredictable. This bit used only in OVSSINGLE, EXTCLKSINGLE and OVSQUAD1X-4X modes.

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	POS		Positive edges on the PCNTn_S0IN inputs are counted in OVSSINGLE mode. Does not invert PCNTn_S1IN input in OVSSINGLE and EXTCLKSINGLE modes
	1	NEG		Negative edges on the PCNTn_S0IN inputs are counted in OVSSINGLE mode. Inverts the PCNTn_S1IN input in OVSSINGLE and EXTCLKSINGLE modes
14	CNTDIR	0	RW	**Non-Quadrature Mode Counter Direction Control**
	The direction of the counter must be set in the OVSSINGLE and EXTCLKSINGLE modes. This bit is ignored in EXTCLKQUAD mode as the direction is automatically detected.			
	Value	Mode		Description
	0	UP		Up counter mode.
	1	DOWN		Down counter mode.
13:12	AUXCNTEV	0x0	RW	**Controls When the Auxiliary Counter Counts**
	Selects whether the auxiliary counter responds to up-count events, down-count events or both			
	Value	Mode		Description
	0	NONE		Never counts.
	1	UP		Counts up on up-count events.
	2	DOWN		Counts up on down-count events.
	3	BOTH		Counts up on both up-count and down-count events.
11:10	CNTEV	0x0	RW	**Controls When the Counter Counts**
	Selects whether the regular counter responds to up-count events, down-count events or both			
	Value	Mode		Description
	0	BOTH		Counts up on up-count and down on down-count events.
	1	UP		Only counts up on up-count events.
	2	DOWN		Only counts down on down-count events.
	3	NONE		Never counts.
9	S1CDIR	0	RW	**Count Direction Determined By S1**
	S1 gives the direction of counting when in the OVSSINGLE or EXTCLKSINGLE modes. When S1 is high, the count direction is given by CNTDIR, and when S1 is low, the count direction is the opposite			
8	HYST	0	RW	**Enable Hysteresis**
	When hysteresis is enabled, the PCNT will always overflow and underflow to TOP/2.			
7	DEBUGHALT	0	RW	**Debug Mode Halt Enable**
	Set to halt the PCNT in debug mode only in OVSSINGLE and OVSQUAD modes. When in EXTCLKSINGLE or EXTCLKQUAD modes, DEBUGHALT does not halt the Pulse Counter.			
	Value			Description
	0			PCNT is running in debug mode.
	1			PCNT is frozen in debug mode.

Bit	Name	Reset	Access	Description
6	AUXCNTRSTEN	0	RW	**Enable AUXCNT Reset**
	The auxiliary counter, AUXCNT, is asynchronously held in reset when this bit is set. The reset is synchronously released two PCNT clock edges after this bit is cleared. If an external clock is used, the reset should be performed by setting and clearing the bit without pending for SYNCBUSY bit.			
5	CNTRSTEN	0	RW	**Enable CNT Reset**
	The counter, CNT, is asynchronously held in reset when this bit is set. The reset is synchronously released two PCNT clock edges after this bit is cleared. If an external clock is used, the reset should be performed by setting and clearing the bit without pending for SYNCBUSY bit. This action clears the counter to its reset value			
4	RSTEN	0	RW	**Enable PCNT Clock Domain Reset**
	The PCNT clock domain is asynchronously held in reset when this bit is set. The reset is synchronously released two PCNT clock edges after this bit is cleared. If an external clock is used, the reset should be performed by setting and clearing the bit without pending for SYNCBUSY bit.			
3	FILT	0	RW	**Enable Digital Pulse Width Filter**
	The filter passes all high and low periods that are at least (FILTLEN+5) clock cycles wide. This filter is only available in OVSSINGLE,OVSQUAD1X-4X modes.			
2:0	MODE	0x0	RW	**Mode Select**
	Selects the mode of operation. The corresponding clock source must be selected from the CMU.			

Value	Mode	Description
0	DISABLE	The module is disabled.
1	OVSSINGLE	Single input LFACLK oversampling mode (available in EM0-EM3).
2	EXTCLKSINGLE	Externally clocked single input counter mode (available in EM0-EM3).
3	EXTCLKQUAD	Externally clocked quadrature decoder mode (available in EM0-EM3).
4	OVSQUAD1X	LFACLK oversampling quadrature decoder 1X mode (available in EM0-EM3).
5	OVSQUAD2X	LFACLK oversampling quadrature decoder 2X mode (available in EM0-EM3).
6	OVSQUAD4X	LFACLK oversampling quadrature decoder 4X mode (available in EM0-EM3).

17.5.2 PCNTn_CMD - Command Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																															0	0
Access																															W1	W1
Name																															LTOPBIM	LCNTIM

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	LTOPBIM	0	W1	**Load TOPB Immediately**
	This bit has no effect since TOPB is not buffered and it is loaded directly into TOP.			
0	LCNTIM	0	W1	**Load CNT Immediately**
	Load PCNTn_TOP into PCNTn_CNT on the next counter clock cycle.			

17.5.3 PCNTn_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																																0
Access																																R
Name																																DIR

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	DIR	0	R	**Current Counter Direction**
	Current direction status of the counter. This bit is valid in EXTCLKQUAD mode only.			

Value	Mode	Description
0	UP	Up counter mode (clockwise in EXTCLKQUAD mode with the EDGE bit in PCNTn_CTRL set to 0).
1	DOWN	Down counter mode.

17.5.4 PCNTn_CNT - Counter Value 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																	0x0000															
Access																	R															
Name																	CNT															

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	CNT	0x0000	R	**Counter Value**
	Gives read access to the counter.			

17.5.5 PCNTn_TOP - Top Value 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																	0x00FF															
Access																	R															
Name																	TOP															

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	TOP	0x00FF	R	**Counter Top Value**
	When counting down, this value is reloaded into PCNTn_CNT when counting past 0. When counting up, 0 is written to the PCNTn_CNT register when counting past this value.			

17.5.6 PCNTn_TOPB - Top Value Buffer Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																	0x00FF															
Access																	RW															
Name																	TOPB															

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	TOPB	0x00FF	RW	**Counter Top Buffer**
	Loaded automatically to TOP when written.			

17.5.7 PCNTn_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																											0	0	0	0	0	0
Access																											R	R	R	R	R	R
Name																											OQSTERR	TCC	AUXOF	DIRCNG	OF	UF

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	OQSTERR	0	R	**Oversampling Quadrature State Error Interrupt**
	Set in the Oversampling Quadrature Mode when incorrect state transition occurs			
4	TCC	0	R	**Triggered Compare Interrupt Read Flag**
	Set upon triggered compare match			
3	AUXOF	0	R	**Auxiliary Overflow Interrupt Read Flag**
	Set when an Auxiliary CNT overflow occurs			
2	DIRCNG	0	R	**Direction Change Detect Interrupt Flag**
	Set when the count direction changes. Set in EXTCLKQUAD mode only.			
1	OF	0	R	**Overflow Interrupt Read Flag**
	Set when a CNT overflow occurs			
0	UF	0	R	**Underflow Interrupt Read Flag**
	Set when a CNT underflow occurs			

17.5.8 PCNTn_IFS - Interrupt Flag Set 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																											0	0	0	0	0	0
Access																											W1	W1	W1	W1	W1	W1
Name																											OQSTERR	TCC	AUXOF	DIRCNG	OF	UF

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	OQSTERR	0	W1	**Set OQSTERR Interrupt Flag**
	Write 1 to set the OQSTERR interrupt flag			
4	TCC	0	W1	**Set TCC Interrupt Flag**
	Write 1 to set the TCC interrupt flag			
3	AUXOF	0	W1	**Set AUXOF Interrupt Flag**
	Write 1 to set the AUXOF interrupt flag			
2	DIRCNG	0	W1	**Set DIRCNG Interrupt Flag**
	Write 1 to set the DIRCNG interrupt flag			
1	OF	0	W1	**Set OF Interrupt Flag**
	Write 1 to set the OF interrupt flag			
0	UF	0	W1	**Set UF Interrupt Flag**
	Write 1 to set the UF interrupt flag			

17.5.9 PCNTn_IFC - Interrupt Flag Clear 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																											0	0	0	0	0	0
Access																											(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name																											OQSTERR	TCC	AUXOF	DIRCNG	OF	UF

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	OQSTERR	0	(R)W1	**Clear OQSTERR Interrupt Flag**
	Write 1 to clear the OQSTERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
4	TCC	0	(R)W1	**Clear TCC Interrupt Flag**
	Write 1 to clear the TCC interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	AUXOF	0	(R)W1	**Clear AUXOF Interrupt Flag**
	Write 1 to clear the AUXOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	DIRCNG	0	(R)W1	**Clear DIRCNG Interrupt Flag**
	Write 1 to clear the DIRCNG interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	OF	0	(R)W1	**Clear OF Interrupt Flag**
	Write 1 to clear the OF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	UF	0	(R)W1	**Clear UF Interrupt Flag**
	Write 1 to clear the UF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

17.5.10 PCNTn_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																											0	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											OQSTERR	TCC	AUXOF	DIRCNG	OF	UF

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	OQSTERR	0	RW	**OQSTERR Interrupt Enable**
	Enable/disable the OQSTERR interrupt			
4	TCC	0	RW	**TCC Interrupt Enable**
	Enable/disable the TCC interrupt			
3	AUXOF	0	RW	**AUXOF Interrupt Enable**
	Enable/disable the AUXOF interrupt			
2	DIRCNG	0	RW	**DIRCNG Interrupt Enable**
	Enable/disable the DIRCNG interrupt			
1	OF	0	RW	**OF Interrupt Enable**
	Enable/disable the OF interrupt			
0	UF	0	RW	**UF Interrupt Enable**
	Enable/disable the UF interrupt			

17.5.11 PCNTn_ROUTELOC0 - I/O Routing Location 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																			0x00								0x00					
Access																			RW								RW					
Name																			S1INLOC								S0INLOC					

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*		
13:8	S1INLOC	0x00	RW	**I/O Location**

Defines the location of the PCNT S1IN input pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

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*		
5:0	S0INLOC	0x00	RW	**I/O Location**

Defines the location of the PCNT S0IN input pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

17.5.12 PCNTn_FREEZE - Freeze 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																																0
Access																																RW
Name																																REGFREEZE

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	REGFREEZE	0	RW	**Register Update Freeze**

When set, the update of the PCNT clock domain is postponed until this bit is cleared. Use this bit to update several registers simultaneously.

Value	Mode	Description
0	UPDATE	Each write access to a PCNT register is updated into the Low Frequency domain as soon as possible.
1	FREEZE	The PCNT clock domain is not updated with the new written value.

17.5.13 PCNTn_SYNCBUSY - Synchronization Busy 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																													0	0	0	0
Access																													R	R	R	R
Name																													OVSCFG	TOPB	CMD	CTRL

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	OVSCFG	0	R	**OVSCFG Register Busy**

Set when the value written to OVSCFG is being synchronized.

2	TOPB	0	R	**TOPB Register Busy**

Set when the value written to TOPB is being synchronized.

1	CMD	0	R	**CMD Register Busy**

Set when the value written to CMD is being synchronized.

0	CTRL	0	R	**CTRL Register Busy**

Set when the value written to CTRL is being synchronized.

17.5.14 PCNTn_AUXCNT - Auxiliary Counter Value 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																	0x0000															
Access																	R															
Name																	AUXCNT															

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	AUXCNT	0x0000	R	**Auxiliary Counter Value**
	Gives read access to the auxiliary counter.			

17.5.15 PCNTn_INPUT - PCNT Input 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																					0		0x0				0		0x0			
Access																					RW		RW				RW		RW			
Name																					S1PRSEN		S1PRSSEL				S0PRSEN		S0PRSSEL			

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	S1PRSEN	0	RW	**S1IN PRS Enable**
	When set, the PRS channel is selected as input to S1IN.			
10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:6	S1PRSSEL	0x0	RW	**S1IN PRS Channel Select**
	Select PRS channel as input to S1IN.			

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected.
1	PRSCH1	PRS Channel 1 selected.
2	PRSCH2	PRS Channel 2 selected.
3	PRSCH3	PRS Channel 3 selected.
4	PRSCH4	PRS Channel 4 selected.
5	PRSCH5	PRS Channel 5 selected.
6	PRSCH6	PRS Channel 6 selected.
7	PRSCH7	PRS Channel 7 selected.
8	PRSCH8	PRS Channel 8 selected.
9	PRSCH9	PRS Channel 9 selected.
10	PRSCH10	PRS Channel 10 selected.
11	PRSCH11	PRS Channel 11 selected.
12	PRSCH12	PRS Channel 12 selected.
13	PRSCH13	PRS Channel 13 selected.
14	PRSCH14	PRS Channel 14 selected.
15	PRSCH15	PRS Channel 15 selected.

5	S0PRSEN	0	RW	**S0IN PRS Enable**
	When set, the PRS channel is selected as input to S0IN.			
4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
3:0	S0PRSSEL	0x0	RW	**S0IN PRS Channel Select**

Select PRS channel as input to S0IN.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected.
1	PRSCH1	PRS Channel 1 selected.
2	PRSCH2	PRS Channel 2 selected.
3	PRSCH3	PRS Channel 3 selected.
4	PRSCH4	PRS Channel 4 selected.
5	PRSCH5	PRS Channel 5 selected.
6	PRSCH6	PRS Channel 6 selected.
7	PRSCH7	PRS Channel 7 selected.
8	PRSCH8	PRS Channel 8 selected.
9	PRSCH9	PRS Channel 9 selected.
10	PRSCH10	PRS Channel 10 selected.
11	PRSCH11	PRS Channel 11 selected.
12	PRSCH12	PRS Channel 12 selected.
13	PRSCH13	PRS Channel 13 selected.
14	PRSCH14	PRS Channel 14 selected.
15	PRSCH15	PRS Channel 15 selected.

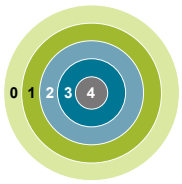
17.5.16 PCNTn_OVSCFG - Oversampling Config Register (Async Reg)

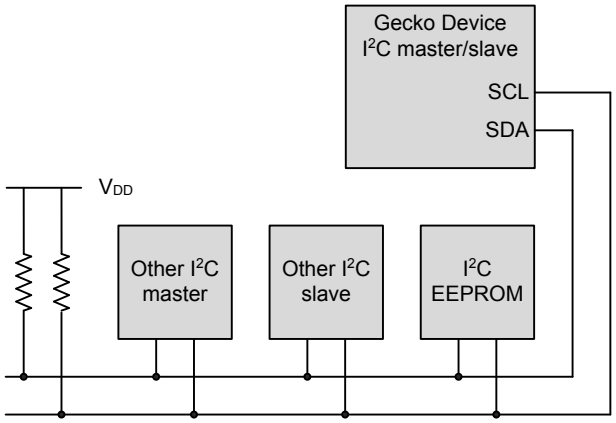
For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																				0					0x00						
Access																				RW					RW						
Name																				FLUTTERRM					FILTLEN						

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	FLUTTERRM	0	RW	**Flutter Remove**
	When set, removes flutter from Quaddecoder inputs S0IN and S1IN. Available only in OVSQUAD1X-4X modes			
11:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	FILTLEN	0x00	RW	**Configure Filter Length for Inputs S0IN and S1IN**
	Used only in OVSINGLE,OVSQUAD1X-4X modes. To use this first enable FILT in PCNTn_CTRL register. Filter length = (FILTLEN + 5) LFACLK cycles			

18. 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.

18.1 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).

18.2 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
  - Active in all energy modes (except EM4)
- 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

18.3 Functional Description

An overview of the I2C module is shown in Figure 18.1 I2C Overview on page 658.

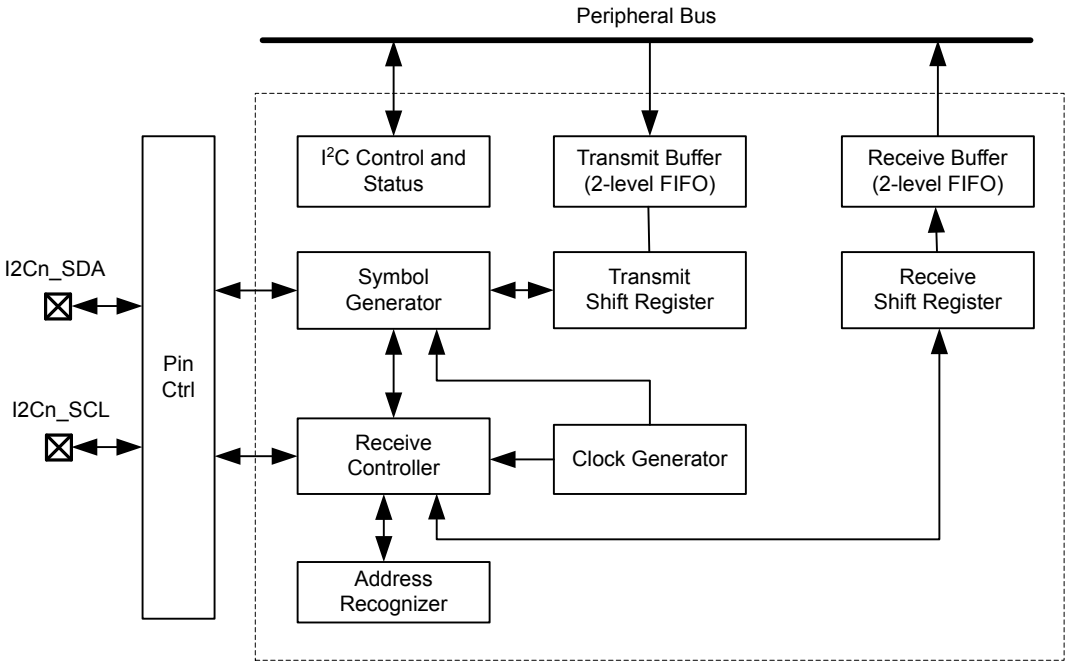


Figure 18.1. I2C Overview

18.3.1 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 18.2 I2C-Bus Example on page 659. 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.

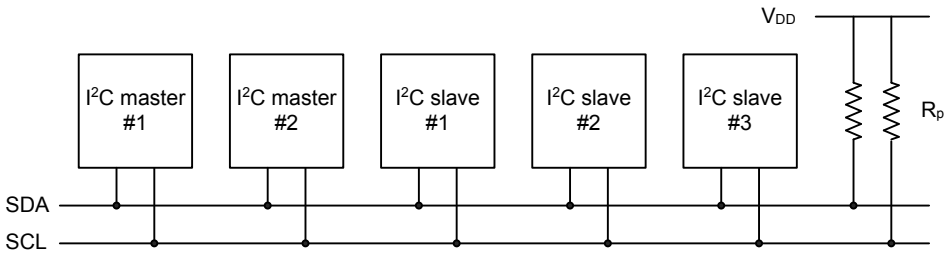


Figure 18.2. I2C-Bus Example

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 18.3 I2C Pull-up Resistor Equation on page 659.

$$Rp(max) = t_r / (0.8473 \times Cb)$$

Figure 18.3. 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 drive strength can be used to control slew rate.
- 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.

18.3.1.1 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 18.4 I2C START and STOP Conditions on page 660, 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.

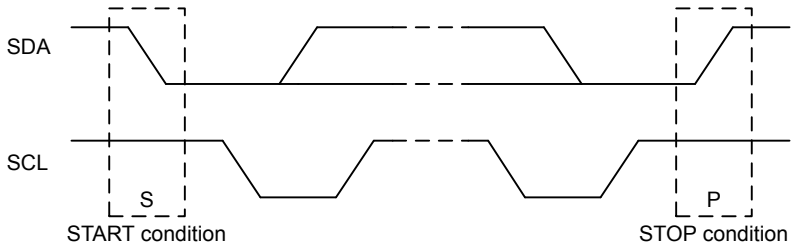


Figure 18.4. I2C START and STOP Conditions

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 18.5 I2C Bit Transfer on I²C-Bus on page 660.

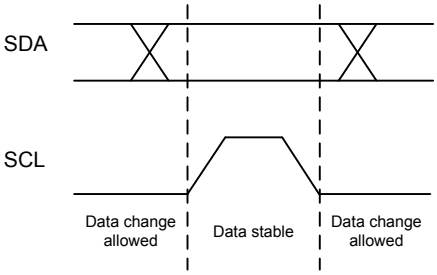


Figure 18.5. I2C Bit Transfer on I²C-Bus

18.3.1.2 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 18.6 I2C Single Byte Write to Slave on page 661, Figure 18.7 I2C Double Byte Read from Slave on page 661, and Figure 18.8 I2C Single Byte Write, then Repeated Start and Single Byte Read on page 661. 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 18.6. I2C Single Byte Write to Slave

Figure 18.7. I2C Double Byte Read from Slave

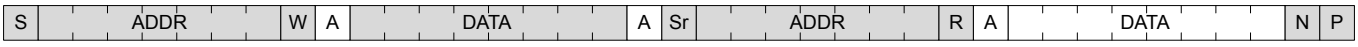


Figure 18.8. I2C Single Byte Write, then Repeated Start and Single Byte Read

18.3.1.3 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 18.1 I2C Reserved I²C Addresses on page 662, and include a General Call address which can be used to broadcast a message to all slaves on the I²C-bus.

Table 18.1. 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

18.3.1.4 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 18.9 I2C Master Transmitter/Slave Receiver with 10-bit Address on page 662.

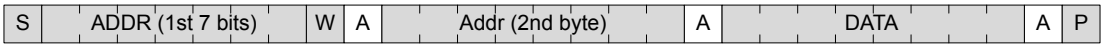


Figure 18.9. I2C Master Transmitter/Slave Receiver with 10-bit Address

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 18.10 I2C Master Receiver/Slave Transmitter with 10-bit Address on page 662.

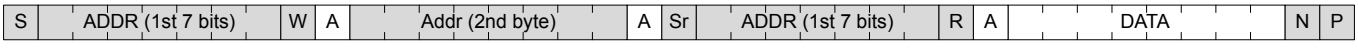


Figure 18.10. I2C Master Receiver/Slave Transmitter with 10-bit Address

18.3.1.5 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^2C module for both masters and slaves. Clock synchronization and clock stretching is also supported.

18.3.2 Enable and Reset

The I^2C is enabled by setting the EN bit in the I2Cn_CTRL register. Whenever this bit is cleared, the internal state of the I^2C is reset, terminating any ongoing transfers.

Note: When enabling the I^2C, the ABORT command or the Bus Idle Timeout feature must be applied prior to use even if the BUSY flag is not set.

18.3.3 Safely Disabling and Changing Slave Configuration

The I^2C 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 I2Cn_CTRL must be reset. This should be done regardless of whether the slave is going to be re-enabled or not.

18.3.4 Clock Generation

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, and is given by the following equation:

$$f_{SCL} = f_{HFPERCCLK}/(((N_{low} + N_{high}) \times (DIV + 1)) + 8),$$

Figure 18.11. I2C Maximum Transmission Rate

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:

$$T_{high} >= ((N_{high}) \times (DIV + 1) + 4)/f_{HFPERCCLK},$$

$$T_{low} >= (N_{low} \times (DIV + 1) + 4)/f_{HFPERCCLK}.$$

Figure 18.12. I2C High and Low Cycles Equations

In worst case, T_{high} and T_{low} can be 1 $f_{HFPERCCLK}$ cycle longer than the number found by above equations due to synchronization uncertainity (i.e., if the synchronization takes 3 $f_{HFPERCCLK}$ 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}). The values of N_{low} and N_{high} and thus the ratio between the high and low parts of the clock signal is controlled by CLHR in the I2Cn_CTRL register.

Note: DIV must be set to 1 during slave mode operation.

18.3.5 Arbitration

Arbitration is enabled by default, but can be disabled by setting the ARBDIS bit in I2Cn_CTRL. When arbitration is enabled, the value on SDA is sensed every time the I2C module attempts to change its value. If the sensed value is different than the value the I2C module tried to output, it is interpreted as a simultaneous transmission by another device, and that the I2C module has lost arbitration.

Whenever arbitration is lost, the ARBLOST interrupt flag in I2Cn_IF is set, any lines held are released, and the I2C device goes idle. If an I2C 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.

18.3.6 Buffers

The I2C peripheral includes separate receive and transmit buffers and shift registers.

18.3.6.1 Transmit Buffer and Shift Register

The I2C transmitter has a 2-level FIFO transmit buffer and a transmit shift register as shown in Figure 18.1 I2C Overview on page 658. A byte is loaded into the transmit buffer by writing to I2Cn_TXDATA or 2 bytes can be loaded simultaneously in the transmit buffer by writing to I2Cn_TXDOUBLE. Figure 18.13 I2C Transmit Buffer Operation on page 664 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 I2Cn_STATUS and the TXC interrupt flags in I2Cn_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.

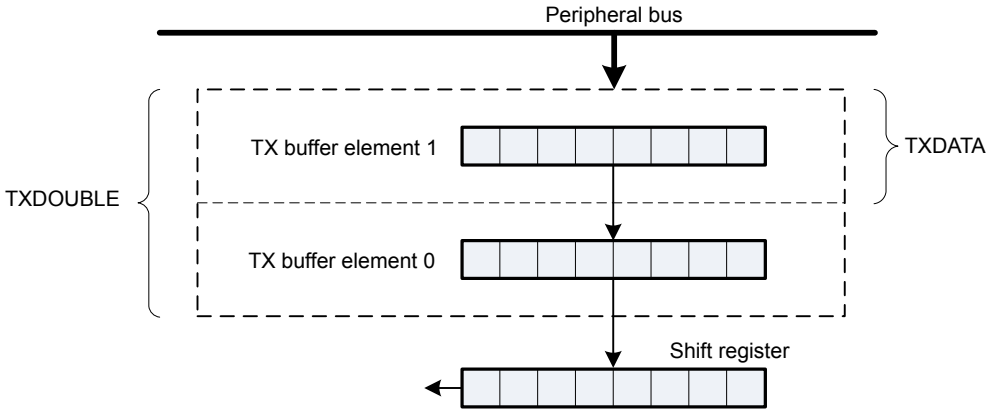


Figure 18.13. I2C Transmit Buffer Operation

The TXBL flags in the I2Cn_STATUS and I2Cn_IF are used to indicate the level of the transmit buffer. TXBIL in I2Cn_CTRL controls the level at which these flag bits are set. If TXBIL is cleared, the flags are set whenever the transmit buffer becomes empty (used when transmitting using I2Cn_TXDOUBLE). If TXBIL is set, the flags are set whenever the transmit buffer goes from full to half-empty or empty (used when transmitting with I2Cn_TXDATA). Both the TXBL status flag and the TXBL interrupt flag are cleared automatically when the condition becomes false.

If an attempt is made to write more bytes to the transmit buffer than the space available, the TXOF interrupt flag in I2Cn_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 I2Cn_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.

18.3.6.2 Receive Buffer and Shift Register

The I²C receiver uses a 2-level FIFO receive buffer and a receive shift register as shown in Figure 18.14 I2C Receive Buffer Operation on page 665. 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.

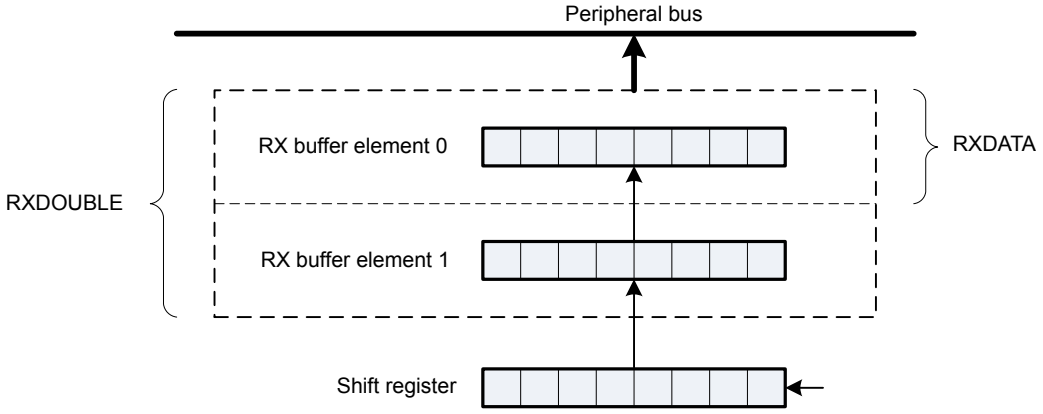


Figure 18.14. I2C Receive Buffer Operation

When a byte becomes available in the receive buffer, the RXDATAV in I2Cn_STATUS and RXDATAV interrupt flag in I2Cn_IF are set. When the buffer becomes full, RXFULL in the I2Cn_STATUS and I2Cn_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 byte.

The data can be fetched from the buffer in two ways. I2Cn_RXDATA gives access to the received byte (if two bytes are received then the one received first is fetched first). I2Cn_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 I2Cn_IF is set to signal the underflow, and the data read from the buffer is undefined.

When using I2Cn_RXDOUBLE to pick data, AUTOACK in I2Cn_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 I2Cn_CMD register.

I2Cn_RXDATAP and I2Cn_RXDOUBLEP can be used to read data from the receive buffer without removing it from the buffer. The RXUF interrupt flag in I2Cn_IF will never be set as a result of reading from I2Cn_RXDATAP and I2Cn_RXDOUBLEP, but the data read through I2Cn_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.

18.3.7 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 I2Cn_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, HFPERCCLK frequency must be higher than 2 MHz for Standard-mode, 9 MHz for Fast-mode, and 20 MHz for Fast-mode Plus.

18.3.7.1 Master State Machine

The master state machine is shown in Figure 18.15 I2C Master State Machine on page 667. 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.

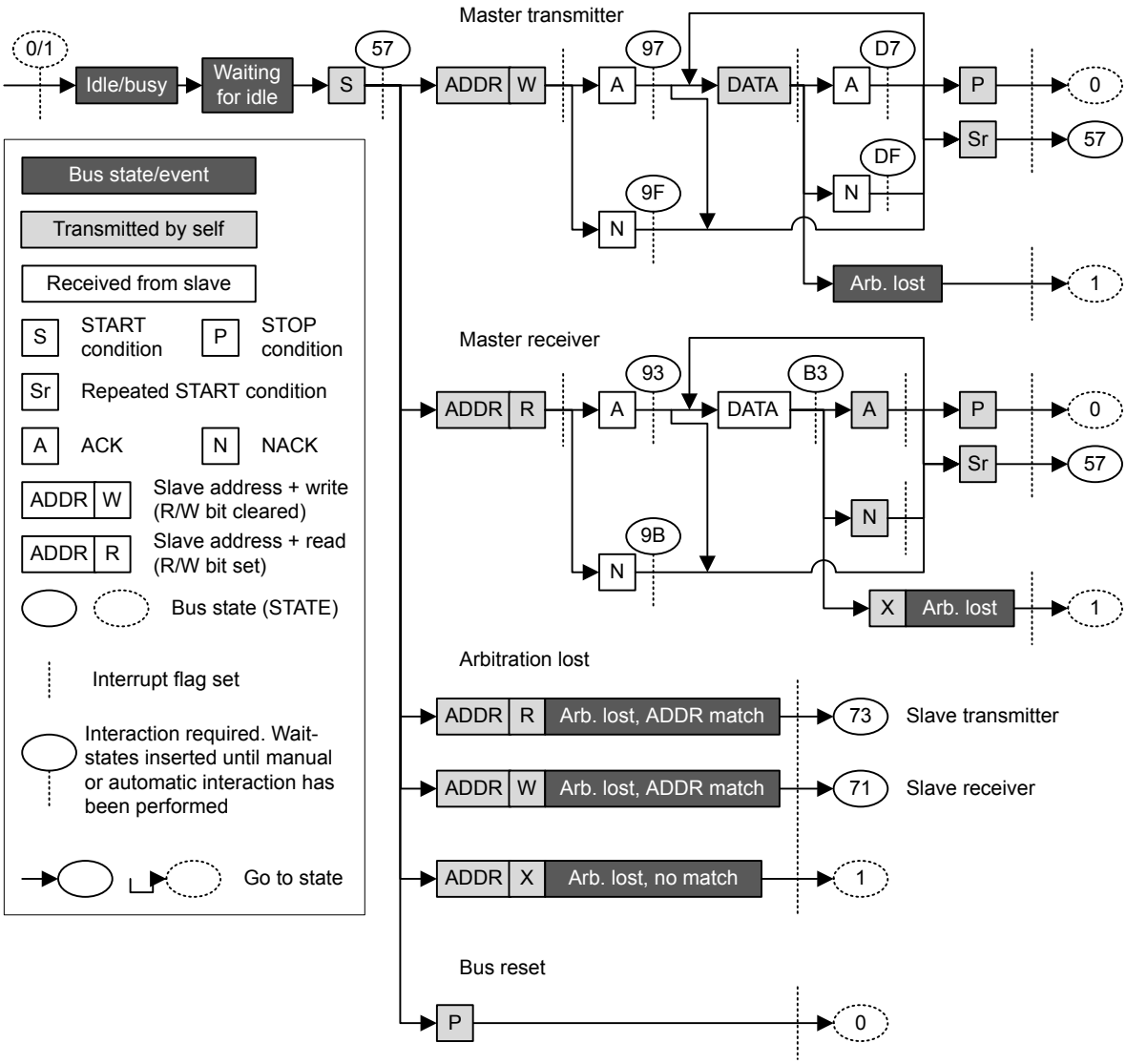


Figure 18.15. I2C Master State Machine

18.3.7.2 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 I2Cn_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 I2Cn_STATE register.

As an example, Table 18.3 I2C Master Transmitter on page 670 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, I2Cn_STATE has a value 0x57, which can be used to identify exactly what the I²C module is waiting for.

Note: The bus would never stop at state 0x57 if the address was available in the transmit buffer.

The different interactions used by the I²C module are listed in Table 18.2 I2C Interactions in Prioritized Order on page 668 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 18.2. I2C Interactions in Prioritized Order

Interaction	Priority	Software action	Automatically continues if
STOP*	1	Set the STOP command bit in I2Cn_CMD	PSTOP is set (STOP pending) in I2Cn_STATUS
ABORT	2	Set the ABORT command bit in I2Cn_CMD	Never, the transmission is aborted
CONT*	3	Set the CONT command bit in I2Cn_CMD	PCONT is set in I2Cn_STATUS (CONT pending)
NACK*	4	Set the NACK command bit in I2Cn_CMD	PNACK is set in I2Cn_STATUS (NACK pending)
ACK*	5	Set the ACK command bit in I2Cn_CMD	AUTOACK is set in I2Cn_CTRL or PACK is set in I2Cn_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 I2Cn_CMD	PSTART is set in I2Cn_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 18.2 I2C Interactions in Prioritized Order on page 668 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 I2Cn_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 18.2 I2C Interactions in Prioritized Order on page 668 is applied to the bus.

Pending commands can be cleared by setting the CLEARPC command bit in I2Cn_CMD.

18.3.7.3 Automatic ACK Interaction

When receiving addresses and data, an ACK command in I2Cn_CMD is normally required after each received byte. When AUTOACK is set in I2Cn_CTRL, an ACK is always pending, and the ACK-pending bit PACK in I2Cn_STATUS is thus always set, even after an ACK has been consumed. This is used when data is picked using I2Cn_RXDOUBLE and can also be used with I2Cn_RXDATA in order to reduce the amount of software interaction required during a transfer.

18.3.7.4 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 I2Cn_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 I2Cn_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 I2Cn_CTRL to an appropriate timeout period and set GIBITO in I2Cn_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.

18.3.7.5 Master Transmitter

To transmit data to a slave, the master must operate as a master transmitter. Table 18.3 I2C Master Transmitter on page 670 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 I2Cn_IF is set, the module is waiting for an interaction, and the bus is frozen. The value of I2Cn_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 I2Cn_IF is set when the I²C module transmits the START.

A master operation is started by issuing a START command by setting START in I2Cn_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 I2Cn_STATE will then be 0x57. 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 I2Cn_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 I2Cn_TXDATA/ I2Cn_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 I2Cn_IF is set. If the arbitration was lost during the transfer of an address, and SLAVE in I2Cn_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 I2Cn_IF is set when a STOP condition is transmitted by the master.

Table 18.3. I2C Master Transmitter

I2Cn_STATE	Description	I2Cn_IF	Required interaction	Response
0x57	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.
0x57	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	

I2Cn_STATE	Description	I2Cn_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

18.3.7.6 Master Receiver

To receive data from a slave, the master must operate as a master receiver, see Table 18.4 I2C Master Receiver on page 672. 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 I2Cn_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 I2Cn_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 I2Cn_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 I2Cn_CMD is set alone, a repeated start condition is transmitted after the ACK/NACK. If STOP in I2Cn_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 I2Cn_IF is set, and the master has a possibility of being selected as a slave given the correct conditions.

Table 18.4. I2C Master Receiver

I2Cn_STATE	Description	I2Cn_IF	Required interaction	Response
0x57	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.
0x57	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	
0x93	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

I2Cn_STATE	Description	I2Cn_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

18.3.8 Bus States

The I2Cn_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 18.5 I2C STATE Values on page 674. When this field is cleared, the I²C module is not a part of any ongoing transmission. The remaining status bits in the I2Cn_STATE register are listed in Table 18.6 I2C Transmission Status on page 674.

Table 18.5. 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 being transmitted
ADDR	3	Address being transmitted or has been received
ADDRACK	4	Address ACK/NACK being transmitted or received
DATA	5	Data being transmitted or received
DATAACK	6	Data ACK/NACK being transmitted or received

Table 18.6. 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: I2Cn_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 I2Cn_STATUS is set.

18.3.9 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 I2Cn_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, HFPERCCLK frequency must be higher than 2 MHz for Standard-mode, 5 MHz for Fast-mode, and 14 MHz for Fast-mode Plus.

18.3.9.1 Slave State Machine

The slave state machine is shown in Figure 18.16 I2C Slave State Machine on page 675. 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.

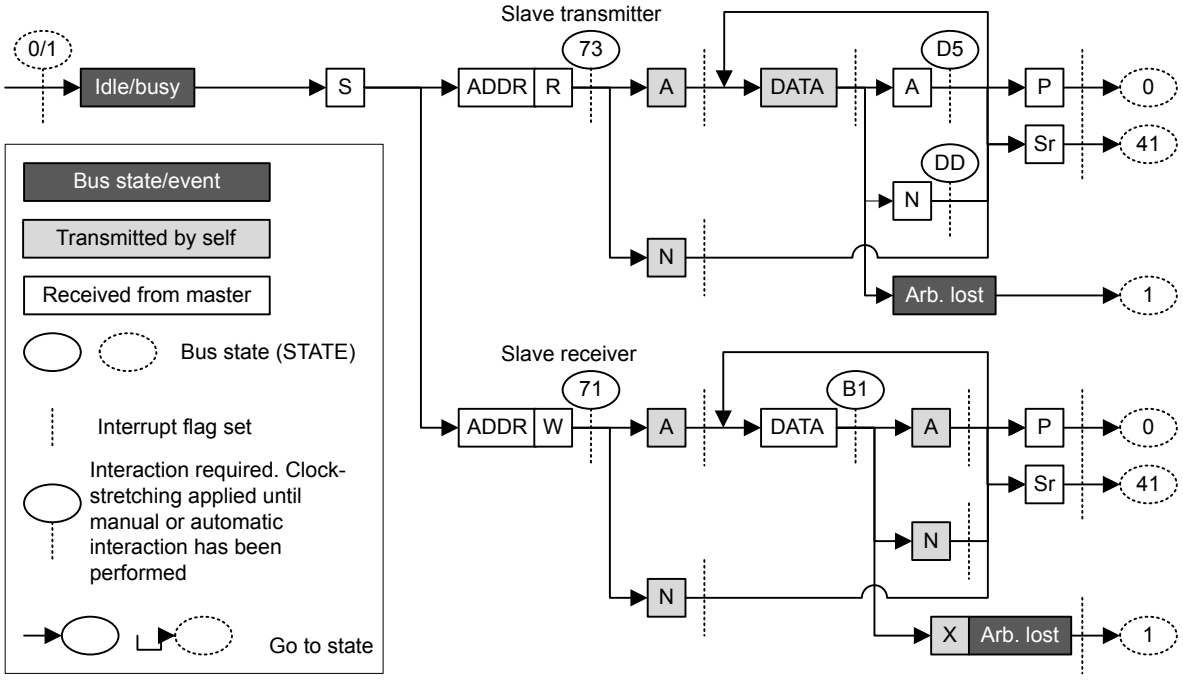


Figure 18.16. I2C Slave State Machine

18.3.9.2 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 18.3.11 Using 10-bit Addresses. Address recognition is supported in all energy modes (except EM4).

The slave address, i.e., the address which the I²C module should be addressed with, is defined in the I2Cn_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 I2Cn_SADDR. The mask is defined in I2Cn_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 I2Cn_SADDR, while a mask 0x70 will match all addresses where the three most significant bits in I2Cn_SADDR and the incoming address are equal.

If GCAMEN in I2Cn_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 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.

18.3.9.3 Slave Transmitter

When SLAVE in I2Cn_CTRL is set, the RSTART interrupt flag in I2Cn_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 I2Cn_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 I2Cn_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 I2Cn_IF will be set regardless of whether the slave is participating in the transmission or not, as long as SLAVE in I2Cn_CTRL is set and a STOP condition is detected.

If arbitration is lost at any time during transmission, the ARBLOST interrupt flag in I2Cn_IF is set, the bus is released and the slave goes idle.

See Table 18.7 I2C Slave Transmitter on page 676 for more information.

Table 18.7. I2C Slave Transmitter

I2Cn_STATE	Description	I2Cn_IF	Required interaction	Response
0x41	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

I2Cn_STATE	Description	I2Cn_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

18.3.9.4 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 I2Cn_IF is not set.

Note: The SSTOP interrupt flag in I2Cn_IF will be set regardless of whether the slave is participating in the transmission or not, as long as SLAVE in I2Cn_CTRL is set and a STOP condition is detected

If arbitration is lost at any time during transmission, the ARBLOST interrupt flag in I2Cn_IF is set, the bus is released and the slave goes idle.

See Table 18.8 I2C - Slave Receiver on page 678 for more information.

Table 18.8. I2C - Slave Receiver

I2Cn_STATE	Description	I2Cn_IF	Required interaction	Response
-	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

18.3.10 Transfer Automation

The I^2C can be set up to complete transfers with a minimal amount of interaction.

18.3.10.1 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.

18.3.10.2 Automatic ACK

When AUTOACK in I2Cn_CTRL is set, an ACK is sent automatically whenever an ACK interaction is possible and no higher priority interactions are pending.

18.3.10.3 Automatic STOP

A STOP can be generated automatically on two conditions. These apply only to the master transmitter.

If AUTOSN in I2Cn_CTRL is set, the I^2C module ends a transmission by transmitting a STOP condition when operating as a master transmitter and a NACK is received.

If AUTOSE in I2Cn_CTRL is set, the I^2C 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.

18.3.11 Using 10-bit Addresses

When using 10-bit addresses in slave mode, set the I2Cn_SADDR register to 1111 0XX where XX are the two most significant bits of the 10-bit address, and set I2Cn_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.

18.3.12 Error Handling

Note: The setting of GCAMEN and SLAVE fields in the I2Cn_CTRL register and the registers I2Cn_SADDR and I2Cn_ROUTELOC0 are considered static. This means that these need to be set before an I^2C transaction starts and need to stay stable during the entire transaction.

18.3.12.1 ABORT Command

Some bus errors may require software intervention to be resolved. The I^2C module provides an ABORT command, which can be set in I2Cn_CMD, to help resolve bus errors.

When the bus for some reason is locked up and the I^2C module is in the middle of a transmission it cannot get out of, or for some other reason the I^2C wants to abort a transmission, the ABORT command can be used.

Setting the ABORT command will make the I^2C 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^2C module forget about any ongoing transfers.

18.3.12.2 Bus Reset

A bus reset can be performed by setting the START and STOP commands in I2Cn_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.

18.3.12.3 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 I2Cn_IF is set to indicate the error, and the module automatically takes a course of action as defined in Table 18.9 I2C Bus Error Response on page 680.

Table 18.9. I2C Bus Error Response

	Misplaced START	Misplaced STOP
In a master/slave operation	Treated as START. Receive address.	Go idle. Perform any pending actions.

18.3.12.4 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 I2C_ROUTE 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 18.3.12.6 Clock Low Timeout

18.3.12.5 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 I2Cn_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 I2Cn_IF is set. The bus can also be set idle automatically on a bus idle timeout. This is enabled by setting GIBITO in I2Cn_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 I2Cn_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 I2Cn_STATUS is set. The timeout can be used to get the I^2C module out of the busy-state it enters when reset, see 18.3.7.4 Reset State.

18.3.12.6 Clock Low Timeout

The clock timeout, which can be configured in CLTO in I2Cn_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 I2Cn_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.

18.3.12.7 Clock Low Error

The I^{2}C 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 I^{2}C module decides to send out a repeated START or a STOP condition while the other device is still sending data. In the I^{2}C protocol specifications, such a combination results in an undefined condition. The I^{2}C deals with this by generating a clock low error. This means that if the I^{2}C is transmitting a repeated START or a STOP condition and another device (another master or a misbehaving slave) pulls SCL low before the I^{2}C sends out the START/STOP condition on SDA, a clock low error is generated. The CLERR interrupt flag is then set in the I2Cn_IF register, any held lines are released and the I^{2}C device goes to idle.

18.3.13 DMA Support

The I^{2}C module has full DMA support. A request for the DMA controller to write to the I^{2}C transmit buffer can come from TXBL (transmit buffer has room for more data). The DMA controller can write to the transmit buffer using the I2Cn_TXDATA or the I2Cn_TXDOUBLE register. In order to write to the I2Cn_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 I2Cn_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 I^{2}C receive buffer can come from RXDATAV (data available in the receive buffer). To receive from I2Cn_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 I2Cn_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.

18.3.14 Interrupts

The interrupts generated by the I^{2}C module are combined into one interrupt vector, I2C_INT. If I^{2}C interrupts are enabled, an interrupt will be made if one or more of the interrupt flags in I2Cn_IF and their corresponding bits in I2Cn_IEN are set.

18.3.15 Wake-up

The I^{2}C 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.

18.4 Register Map

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

Offset	Name	Type	Description
0x000	I2Cn_CTRL	RW	Control Register
0x004	I2Cn_CMD	W1	Command Register
0x008	I2Cn_STATE	R	State Register
0x00C	I2Cn_STATUS	R	Status Register
0x010	I2Cn_CLKDIV	RW	Clock Division Register
0x014	I2Cn_SADDR	RW	Slave Address Register
0x018	I2Cn_SADDRMASK	RW	Slave Address Mask Register
0x01C	I2Cn_RXDATA	R(a)	Receive Buffer Data Register
0x020	I2Cn_RXDOUBLE	R(a)	Receive Buffer Double Data Register
0x024	I2Cn_RXDATAP	R	Receive Buffer Data Peek Register
0x028	I2Cn_RXDOUBLEP	R	Receive Buffer Double Data Peek Register
0x02C	I2Cn_TXDATA	W	Transmit Buffer Data Register
0x030	I2Cn_TXDOUBLE	W	Transmit Buffer Double Data Register
0x034	I2Cn_IF	R	Interrupt Flag Register
0x038	I2Cn_IFS	W1	Interrupt Flag Set Register
0x03C	I2Cn_IFC	(R)W1	Interrupt Flag Clear Register
0x040	I2Cn_IEN	RW	Interrupt Enable Register
0x044	I2Cn_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x048	I2Cn_ROUTELOC0	RW	I/O Routing Location Register

18.5 Register Description

18.5.1 I2Cn_CTRL - 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			0		0x0				0x0		0	0	0	0	0	0	0	0
Access														RW			RW		RW				RW		RW	RW	RW	RW	RW	RW	RW	RW
Name														CLTO			GIBITO		BITO				CLHR		TXBIL	GCAMEN	ARBDIS	AUTOSN	AUTOSE	AUTOACK	SLAVE	EN

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*		
18:16	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

$$timeout = PCC/(f_{SCL} \times (N_{low} + N_{high}))$$

Value	Mode	Description
0	OFF	Timeout disabled
1	40PCC	Timeout after 40 prescaled clock cycles. In standard mode at 100 kHz, this results in a 50us timeout.
2	80PCC	Timeout after 80 prescaled clock cycles. In standard mode at 100 kHz, this results in a 100us timeout.
3	160PCC	Timeout after 160 prescaled clock cycles. In standard mode at 100 kHz, this results in a 200us timeout.
4	320PCC	Timeout after 320 prescaled clock cycles. In standard mode at 100 kHz, this results in a 400us timeout.
5	1024PCC	Timeout after 1024 prescaled clock cycles. In standard mode at 100 kHz, this results in a 1280us timeout.

Bit	Name	Reset	Access	Description
15	GIBITO	0	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	Description
0	A bus idle timeout has no effect on the bus state.
1	A bus idle timeout tells the I^2C module that the bus is idle, allowing new transfers to be initiated.

Bit	Name	Reset	Access	Description
14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
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

$$timeout = PCC/(f_{SCL} \times (N_{low} + N_{high}))$$

Value	Mode	Description
0	OFF	Timeout disabled
1	40PCC	Timeout after 40 prescaled clock cycles. In standard mode at 100 kHz, this results in a 50us timeout.
2	80PCC	Timeout after 80 prescaled clock cycles. In standard mode at 100 kHz, this results in a 100us timeout.
3	160PCC	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 ratio between the low and high parts of the clock signal generated on SCL as master.

Value	Mode	Description
0	STANDARD	The ratio between low period and high period counters (N_{low}:N_{high}) is 4:4
1	ASYMMETRIC	The ratio between low period and high period counters (N_{low}:N_{high}) is 6:3
2	FAST	The ratio between low period and high period counters (N_{low}:N_{high}) is 11:6

7	TXBIL	0	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 non-empty.
1	HALFFULL	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	0	RW	**General Call Address Match Enable**

Set to enable address match on general call in addition to the programmed slave address.

Value	Description
0	General call address will be NACK'ed if it is not included by the slave address and address mask.

Bit	Name	Reset	Access	Description
	1			When a general call address is received, a software response is required.
5	ARBDIS	0	RW	**Arbitration Disable**
	A master or slave will not release the bus upon losing arbitration.			
	Value			Description
	0			When a device loses arbitration, the ARB interrupt flag is set and the bus is released.
	1			When a device loses arbitration, the ARB interrupt flag is set, but communication proceeds.
4	AUTOSN	0	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			Description
	0			Stop is not automatically sent if a NACK is received from a slave.
	1			The master automatically sends a STOP if a NACK is received from a slave.
3	AUTOSE	0	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			Description
	0			A stop must be sent manually when no more data is to be transmitted.
	1			The master automatically sends a STOP when no more data is available for transmission.
2	AUTOACK	0	RW	**Automatic Acknowledge**
	Set to enable automatic acknowledges.			
	Value			Description
	0			Software must give one ACK command for each ACK transmitted on the I^2C bus.
	1			Addresses that are not automatically NACK'ed, and all data is automatically acknowledged.
1	SLAVE	0	RW	**Addressable as Slave**
	Set this bit to allow the device to be selected as an I^2C slave.			
	Value			Description
	0			All addresses will be responded to with a NACK
	1			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.
0	EN	0	RW	**I^2C Enable**
	Use this bit to enable or disable the I^2C module.			

Bit	Name	Reset	Access	Description
	Value			Description
	0			The I^2C module is disabled. And its internal state is cleared
	1			The I^2C module is enabled.

18.5.2 I2Cn_CMD - Command 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																									0	0	0	0	0	0	0	0
Access																									W1	W1	W1	W1	W1	W1	W1	W1
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	0	W1	**Clear Pending Commands**
	Set to clear pending commands.			
6	CLEARTX	0	W1	**Clear TX**
	Set to clear transmit buffer and shift register. Will not abort ongoing transfer.			
5	ABORT	0	W1	**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	0	W1	**Continue Transmission**
	Set to continue transmission after a NACK has been received.			
3	NACK	0	W1	**Send NACK**
	Set to transmit a NACK the next time an acknowledge is required.			
2	ACK	0	W1	**Send ACK**
	Set to transmit an ACK the next time an acknowledge is required.			
1	STOP	0	W1	**Send Stop Condition**
	Set to send stop condition as soon as possible.			
0	START	0	W1	**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.			

18.5.3 I2Cn_STATE - State 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			0	0	0	0	1
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 transmitted or received
3	ADDR	Address transmitted or received
4	ADDRACK	Address ack/nack transmitted or received
5	DATA	Data transmitted or received
6	DATAACK	Data ack/nack transmitted or received

4	BUSHOLD	0	R	**Bus Held** Set if the bus is currently being held by this I²C module.
3	NACKED	0	R	**Nack Received** Set if a NACK was received and STATE is ADDRACK or DATAACK.
2	TRANSMITTER	0	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	0	R	**Master** Set when operating as an I²C master. When cleared, the system may be operating as an I²C slave.
0	BUSY	1	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.

18.5.4 I2Cn_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																							0	0	1	0	0	0	0	0	0	0
Access																							R	R	R	R	R	R	R	R	R	R
Name																							RXFULL	RXDATAV	TXBL	TXC	PABORT	PCONT	PNACK	PACK	PSTOP	PSTART

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	RXFULL	0	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	0	R	**RX Data Valid**
	Set when data is available in the receive buffer. Cleared when the receive buffer is empty.			
7	TXBL	1	R	**TX Buffer Level**
	Indicates the level of the transmit buffer. Set when the transmit buffer is empty, and cleared when it is full.			
6	TXC	0	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	0	R	**Pending Abort**
	An abort is pending and will be transmitted as soon as possible.			
4	PCONT	0	R	**Pending Continue**
	A continue is pending and will be transmitted as soon as possible.			
3	PNACK	0	R	**Pending NACK**
	A not-acknowledge is pending and will be transmitted as soon as possible.			
2	PACK	0	R	**Pending ACK**
	An acknowledge is pending and will be transmitted as soon as possible.			
1	PSTOP	0	R	**Pending STOP**
	A stop condition is pending and will be transmitted as soon as possible.			
0	PSTART	0	R	**Pending START**
	A start condition is pending and will be transmitted as soon as possible.			

18.5.5 I2Cn_CLKDIV - Clock Division 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																												0x000				
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	0x000	RW	**Clock Divider**
	Specifies the clock divider for the I^{2}C. Note that DIV must be 1 or higher when slave is enabled.			

18.5.6 I2Cn_SADDR - Slave Address 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																												0x00				
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	0x00	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*		

18.5.7 I2Cn_SADDRMASK - Slave Address Mask 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																									0x00							
Access																									RW							
Name																									MASK							

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	MASK	0x00	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*		

18.5.8 I2Cn_RXDATA - Receive Buffer Data Register (Actionable Reads)

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																									0x00							
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	0x00	R	**RX Data**
	Use this register to read from the receive buffer. Buffer is emptied on read access.			

18.5.9 I2Cn_RXDOUBLE - Receive Buffer Double Data Register (Actionable Reads)

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																	0x00								0x00							
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	0x00	R	**RX Data 1**
	Second byte read from buffer. Buffer is emptied on read access.			
7:0	RXDATA0	0x00	R	**RX Data 0**
	First byte read from buffer. Buffer is emptied on read access.			

18.5.10 I2Cn_RXDATAP - Receive Buffer Data Peek 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																									0x00							
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	0x00	R	**RX Data Peek**
	Use this register to read from the receive buffer. Buffer is not emptied on read access.			

18.5.11 I2Cn_RXDOUBLEP - Receive Buffer Double Data Peek 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																	0x00								0x00							
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	0x00	R	**RX Data 1 Peek**
	Second byte read from buffer. Buffer is not emptied on read access.			
7:0	RXDATAP0	0x00	R	**RX Data 0 Peek**
	First byte read from buffer. Buffer is not emptied on read access.			

18.5.12 I2Cn_TXDATA - Transmit Buffer 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																									0x00							
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	0x00	W	**TX Data**
	Use this register to write a byte to the transmit buffer.			

18.5.13 I2Cn_TXDOUBLE - Transmit Buffer Double Data 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																	0x00								0x00							
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	0x00	W	**TX Data**
	Second byte to write to buffer.			
7:0	TXDATA0	0x00	W	**TX Data**
	First byte to write to buffer.			

18.5.14 I2Cn_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														0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Access														R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name														CLERR	RXFULL	SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK	RXDATAV	TXBL	TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	CLERR	0	R	**Clock Low Error Interrupt Flag**
	Set when the clock is pulled low before a START or a STOP condition could be transmitted.			
17	RXFULL	0	R	**Receive Buffer Full Interrupt Flag**
	Set when the receive buffer becomes full.			
16	SSTOP	0	R	**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	0	R	**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	0	R	**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	0	R	**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	0	R	**Transmit Buffer Overflow Interrupt Flag**
	Set when data is written to the transmit buffer while the transmit buffer is full.			
11	BUSHOLD	0	R	**Bus Held Interrupt Flag**
	Set when the bus becomes held by the I^2C module.			
10	BUSERR	0	R	**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	0	R	**Arbitration Lost Interrupt Flag**
	Set when arbitration is lost.			
8	MSTOP	0	R	**Master STOP Condition Interrupt Flag**
	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	0	R	**Not Acknowledge Received Interrupt Flag**
	Set when a NACK has been received.			
6	ACK	0	R	**Acknowledge Received Interrupt Flag**
	Set when an ACK has been received.			

Bit	Name	Reset	Access	Description
5	RXDATAV	0	R	**Receive Data Valid Interrupt Flag**
	Set when data is available in the receive buffer. Cleared automatically when the receive buffer is read.			
4	TXBL	1	R	**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	0	R	**Transfer Completed Interrupt Flag**
	Set when the transmit shift register becomes empty and there is no more data in the transmit buffer.			
2	ADDR	0	R	**Address Interrupt Flag**
	Set when incoming address is accepted, i.e. own address or general call address is received.			
1	RSTART	0	R	**Repeated START Condition Interrupt Flag**
	Set when a repeated start condition is detected.			
0	START	0	R	**START Condition Interrupt Flag**
	Set when a start condition is successfully transmitted.			

18.5.15 I2Cn_IFS - Interrupt Flag Set 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														0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0
Access														W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1			W1	W1	W1	W1
Name														CLERR	RXFULL	SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK			TXC	ADDR	RSTART	START

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	CLERR	0	W1	**Set CLERR Interrupt Flag**
	Write 1 to set the CLERR interrupt flag			
17	RXFULL	0	W1	**Set RXFULL Interrupt Flag**
	Write 1 to set the RXFULL interrupt flag			
16	SSTOP	0	W1	**Set SSTOP Interrupt Flag**
	Write 1 to set the SSTOP interrupt flag			
15	CLTO	0	W1	**Set CLTO Interrupt Flag**
	Write 1 to set the CLTO interrupt flag			
14	BITO	0	W1	**Set BITO Interrupt Flag**
	Write 1 to set the BITO interrupt flag			
13	RXUF	0	W1	**Set RXUF Interrupt Flag**
	Write 1 to set the RXUF interrupt flag			
12	TXOF	0	W1	**Set TXOF Interrupt Flag**
	Write 1 to set the TXOF interrupt flag			
11	BUSHOLD	0	W1	**Set BUSHOLD Interrupt Flag**
	Write 1 to set the BUSHOLD interrupt flag			
10	BUSERR	0	W1	**Set BUSERR Interrupt Flag**
	Write 1 to set the BUSERR interrupt flag			
9	ARBLOST	0	W1	**Set ARBLOST Interrupt Flag**
	Write 1 to set the ARBLOST interrupt flag			
8	MSTOP	0	W1	**Set MSTOP Interrupt Flag**
	Write 1 to set the MSTOP interrupt flag			
7	NACK	0	W1	**Set NACK Interrupt Flag**
	Write 1 to set the NACK interrupt flag			
6	ACK	0	W1	**Set ACK Interrupt Flag**
	Write 1 to set the ACK interrupt flag			
5:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
3	TXC	0	W1	**Set TXC Interrupt Flag**
	Write 1 to set the TXC interrupt flag			
2	ADDR	0	W1	**Set ADDR Interrupt Flag**
	Write 1 to set the ADDR interrupt flag			
1	RSTART	0	W1	**Set RSTART Interrupt Flag**
	Write 1 to set the RSTART interrupt flag			
0	START	0	W1	**Set START Interrupt Flag**
	Write 1 to set the START interrupt flag			

18.5.16 I2Cn_IFC - Interrupt Flag Clear 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														0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0
Access														(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1			(R)W1	(R)W1	(R)W1	(R)W1
Name														CLERR	RXFULL	SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK			TXC	ADDR	RSTART	START

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	CLERR	0	(R)W1	**Clear CLERR Interrupt Flag**
	Write 1 to clear the CLERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
17	RXFULL	0	(R)W1	**Clear RXFULL Interrupt Flag**
	Write 1 to clear the RXFULL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
16	SSTOP	0	(R)W1	**Clear SSTOP Interrupt Flag**
	Write 1 to clear the SSTOP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
15	CLTO	0	(R)W1	**Clear CLTO Interrupt Flag**
	Write 1 to clear the CLTO interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
14	BITO	0	(R)W1	**Clear BITO Interrupt Flag**
	Write 1 to clear the BITO interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
13	RXUF	0	(R)W1	**Clear RXUF Interrupt Flag**
	Write 1 to clear the RXUF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
12	TXOF	0	(R)W1	**Clear TXOF Interrupt Flag**
	Write 1 to clear the TXOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
11	BUSHOLD	0	(R)W1	**Clear BUSHOLD Interrupt Flag**
	Write 1 to clear the BUSHOLD interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
10	BUSERR	0	(R)W1	**Clear BUSERR Interrupt Flag**
	Write 1 to clear the BUSERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	ARBLOST	0	(R)W1	**Clear ARBLOST Interrupt Flag**
	Write 1 to clear the ARBLOST interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
8	MSTOP	0	(R)W1	**Clear MSTOP Interrupt Flag**
	Write 1 to clear the MSTOP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	NACK	0	(R)W1	**Clear NACK Interrupt Flag**
	Write 1 to clear the NACK interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
6	ACK	0	(R)W1	**Clear ACK Interrupt Flag**
	Write 1 to clear the ACK interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
5:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	TXC	0	(R)W1	**Clear TXC Interrupt Flag**
	Write 1 to clear the TXC interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	ADDR	0	(R)W1	**Clear ADDR Interrupt Flag**
	Write 1 to clear the ADDR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	RSTART	0	(R)W1	**Clear RSTART Interrupt Flag**
	Write 1 to clear the RSTART interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	START	0	(R)W1	**Clear START Interrupt Flag**
	Write 1 to clear the START interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

18.5.17 I2Cn_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														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access														RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name														CLERR	RXFULL	SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK	RXDATAV	TXBL	TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18	CLERR	0	RW	**CLERR Interrupt Enable**
	Enable/disable the CLERR interrupt			
17	RXFULL	0	RW	**RXFULL Interrupt Enable**
	Enable/disable the RXFULL interrupt			
16	SSTOP	0	RW	**SSTOP Interrupt Enable**
	Enable/disable the SSTOP interrupt			
15	CLTO	0	RW	**CLTO Interrupt Enable**
	Enable/disable the CLTO interrupt			
14	BITO	0	RW	**BITO Interrupt Enable**
	Enable/disable the BITO interrupt			
13	RXUF	0	RW	**RXUF Interrupt Enable**
	Enable/disable the RXUF interrupt			
12	TXOF	0	RW	**TXOF Interrupt Enable**
	Enable/disable the TXOF interrupt			
11	BUSHOLD	0	RW	**BUSHOLD Interrupt Enable**
	Enable/disable the BUSHOLD interrupt			
10	BUSERR	0	RW	**BUSERR Interrupt Enable**
	Enable/disable the BUSERR interrupt			
9	ARBLOST	0	RW	**ARBLOST Interrupt Enable**
	Enable/disable the ARBLOST interrupt			
8	MSTOP	0	RW	**MSTOP Interrupt Enable**
	Enable/disable the MSTOP interrupt			
7	NACK	0	RW	**NACK Interrupt Enable**
	Enable/disable the NACK interrupt			
6	ACK	0	RW	**ACK Interrupt Enable**
	Enable/disable the ACK interrupt			

Bit	Name	Reset	Access	Description
5	RXDATAV	0	RW	**RXDATAV Interrupt Enable**
	Enable/disable the RXDATAV interrupt			
4	TXBL	0	RW	**TXBL Interrupt Enable**
	Enable/disable the TXBL interrupt			
3	TXC	0	RW	**TXC Interrupt Enable**
	Enable/disable the TXC interrupt			
2	ADDR	0	RW	**ADDR Interrupt Enable**
	Enable/disable the ADDR interrupt			
1	RSTART	0	RW	**RSTART Interrupt Enable**
	Enable/disable the RSTART interrupt			
0	START	0	RW	**START Interrupt Enable**
	Enable/disable the START interrupt			

18.5.18 I2Cn_ROUTEPEN - I/O Routing Pin Enable 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																															0	0
Access																															RW	RW
Name																															SCLPEN	SDAPEN

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	SCLPEN	0	RW	**SCL Pin Enable**
	When set, the SCL pin of the I^2C is enabled.			
0	SDAPEN	0	RW	**SDA Pin Enable**
	When set, the SDA pin of the I^2C is enabled.			

18.5.19 I2Cn_ROUTELOC0 - I/O Routing Location 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																			0x00								0x00					
Access																			RW								RW					
Name																			SCLLOC								SDALOC					

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*		
13:8	SCLLOC	0x00	RW	**I/O Location**

Decides the location of the I²C SCL pin.

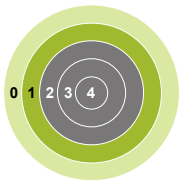
Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

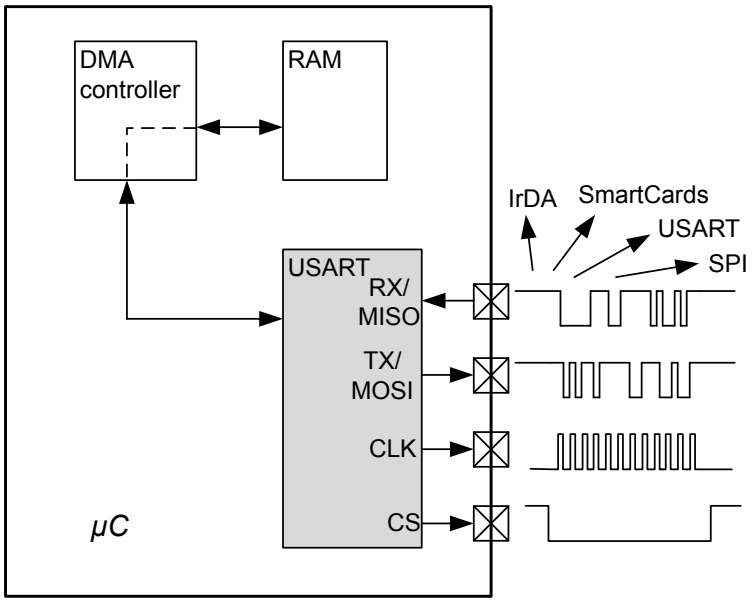
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	SDALOC	0x00	RW	**I/O Location**

Decides the location of the I²C SDA pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

19. 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.

19.1 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.

19.2 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 ($HFPERCLK_{USARTn}$)
- 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

19.3 Functional Description

An overview of the USART module is shown in Figure 19.1 USART Overview on page 705.

This section describes all possible USART features. Refer to the device data sheet to see what features a specific USART instance supports.

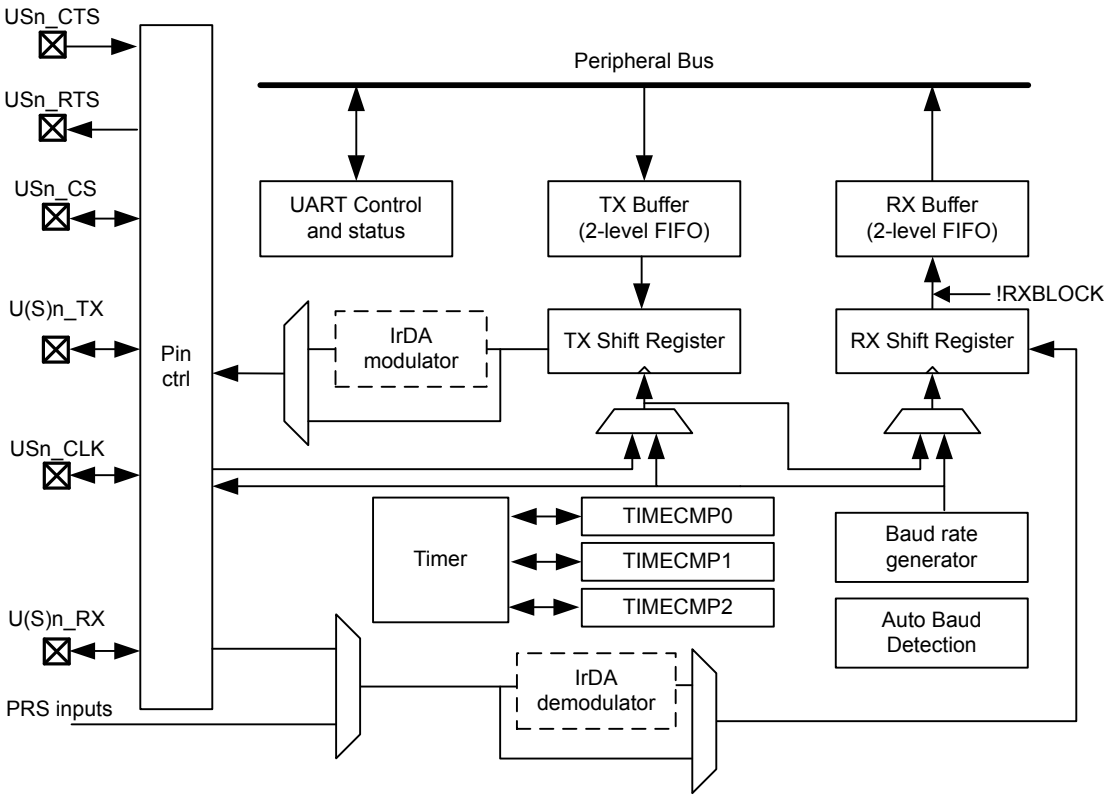


Figure 19.1. USART Overview

19.3.1 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 19.1 USART Asynchronous Vs. Synchronous Mode on page 706. Full duplex and half duplex communication is supported in both asynchronous and synchronous mode.

Table 19.1. 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 19.2 USART Pin Usage on page 706 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 19.3.2.14 Local Loopback and 19.3.3.3 Master Mode respectively.

Table 19.2. 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]

19.3.2 Asynchronous Operation

The USART operates in asynchronous mode when SYNC in USARTn_CTRL is cleared to 0.

19.3.2.1 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 19.2 USART Asynchronous Frame Format on page 707.

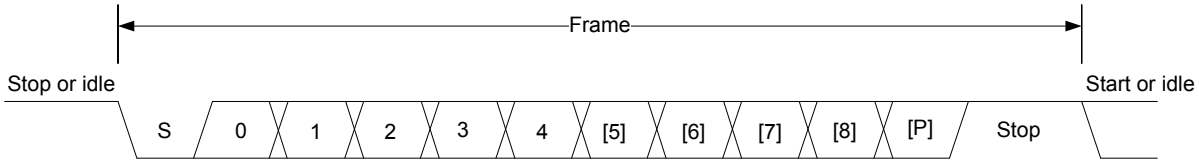


Figure 19.2. USART Asynchronous Frame Format

The number of data bits in a frame is set by DATABITS in USARTn_FRAME, see Table 19.3 USART Data Bits on page 707, and the number of stop-bits is set by STOPBITS in USARTn_FRAME, see Table 19.4 USART Stop Bits on page 707. 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 19.3. 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 19.4. 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.

19.3.2.2 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 19.5 USART Parity Bits on page 708. 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 19.5. USART Parity Bits

PARITY BITS [1:0]	Description
00	No parity bit (Default)
01	Reserved
10	Even parity
11	Odd parity

19.3.2.3 Clock Generation

Note: Not all USART instances are using the same peripheral clock. Normally the USART uses $HFPERCLK_{USARTn}$, however USART2 supports higher frequencies and therefore uses $HFPERBCLK_{USART2}$. This chapter describes the general case and therefore uses $HFPERCLK_{USARTn}$ and $f_{HFPERCLK}$, which should be interpreted as $HFPERBCLK_{USARTn}$ and $f_{HFPERBCLK}$ for USART2. 10.3.1.4 HFPERCLK, HFPERBCLK, HFPERCCLK - High Frequency Peripheral Clocks shows which peripheral uses what peripheral clock.

The USART clock defines the transmission and reception data rate. When operating in asynchronous mode, the baud rate (bit-rate) is given by Figure 19.3 USART Baud Rate on page 709.

$$br = f_{HFPERCLK}/(oversample \times (1 + USARTn\_CLKDIV/256))$$

Figure 19.3. USART Baud Rate

where $f_{HFPERCLK}$ is the peripheral clock ($HFPERCLK_{USARTn}$) frequency and oversample is the oversampling rate as defined by OVS in USARTn_CTRL, see Table 19.6 USART Oversampling on page 709.

Table 19.6. 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.

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 USARTn_CLKDIV can be calculated by using Figure 19.4 USART Desired Baud Rate on page 709:

$$USARTn\_CLKDIV = 256 \times (f_{HFPERCLK}/(oversample \times brdesired) - 1)$$

Figure 19.4. USART Desired Baud Rate

Table 19.7 USART Baud Rates @ 4MHz Peripheral Clock With 20 Bit CLKDIV on page 709 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 19.7. 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

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 %
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
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

19.3.2.4 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.

19.3.2.5 Data Transmission

Asynchronous data transmission is initiated by writing data to the transmit buffer using one of the methods described in 19.3.2.6 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.

19.3.2.6 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 19.5 USART Transmit Buffer Operation on page 711 shows the basics of the transmit buffer when DATABITS in USARTn_FRAME is configured to less than 10 bits.

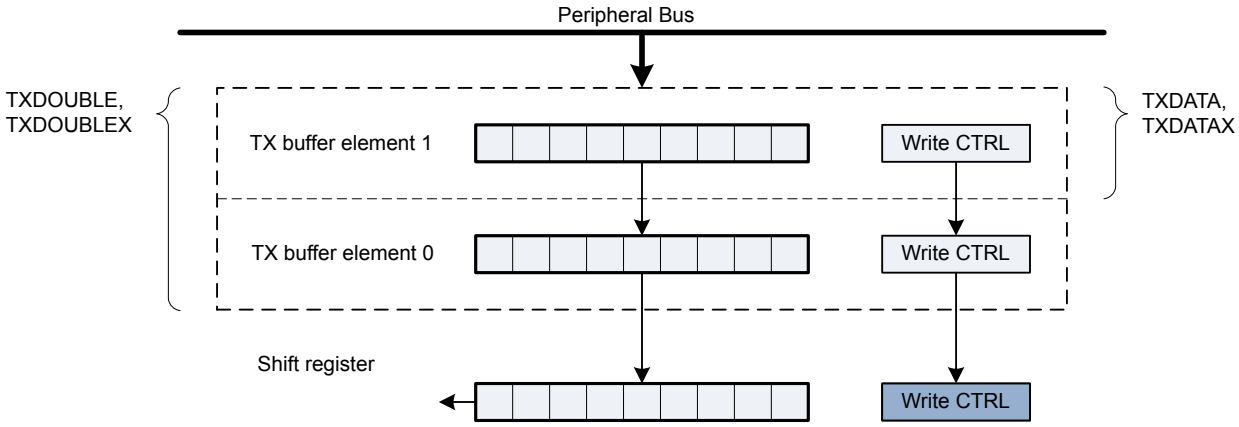


Figure 19.5. USART Transmit Buffer Operation

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.

There is a TXIDLE status bit in USARTn_STATUS to provide an indication of when the transmitter is idle. The combined count of TX buffer element 0, TX buffer element 1, and TX shift register is called TXBUFCNT in USARTn_STATUS. For large frames, the count is only of TX buffer entry 0 and the TX shifter register.

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.

19.3.2.7 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.

19.3.2.8 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.

19.3.2.9 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 19.6 USART Receive Buffer Operation on page 713.

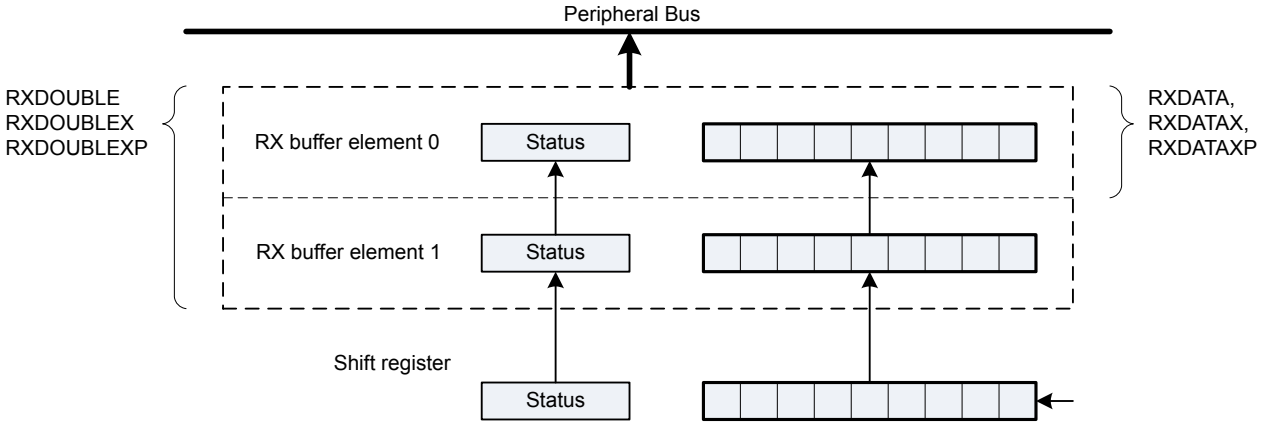


Figure 19.6. USART Receive Buffer Operation

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.

19.3.2.10 Blocking Incoming Data

When using hardware frame recognition, as detailed in 19.3.2.20 Multi-Processor Mode and 19.3.2.21 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 19.3.2.20 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.

19.3.2.11 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 19.7 USART Sampling of Start and Data Bits on page 715. 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 19.7 USART Sampling of Start and Data Bits on page 715.

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.

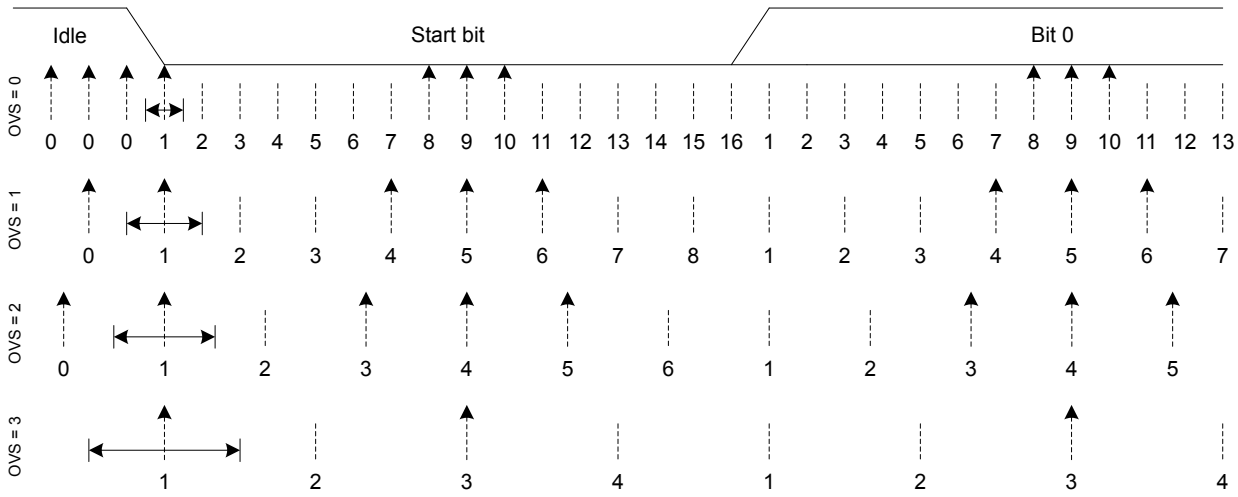


Figure 19.7. USART Sampling of Start and Data Bits

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 19.8 USART Sampling of Stop Bits when Number of Stop Bits are 1 or More on page 716. 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 19.8 USART Sampling of Stop Bits when Number of Stop Bits are 1 or More on page 716, 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.

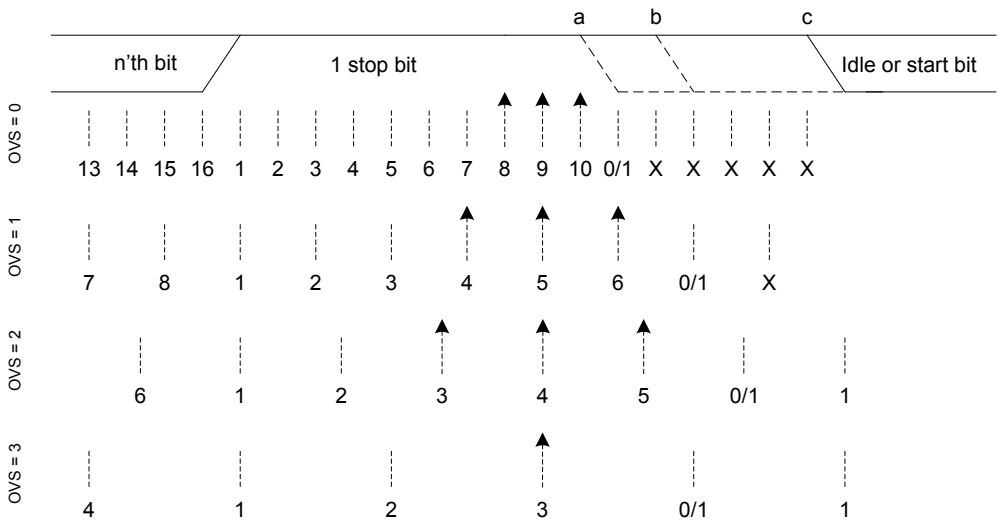


Figure 19.8. USART Sampling of Stop Bits when Number of Stop Bits are 1 or More

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.

19.3.2.12 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.

19.3.2.13 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.

19.3.2.14 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 19.9 USART Local Loopback on page 717. 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.

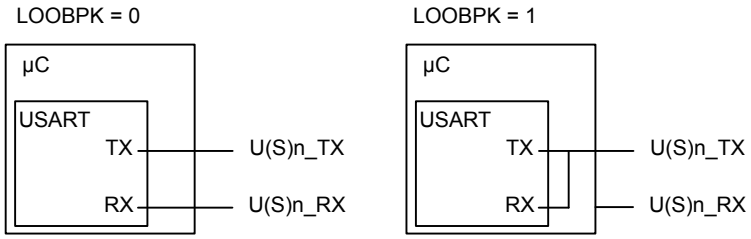


Figure 19.9. USART Local Loopback

19.3.2.15 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.

19.3.2.16 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, 19.3.10 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,19.3.4 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.

19.3.2.17 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, 19.3.10 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.

Figure 19.10 USART Half Duplex Communication with External Driver on page 718 shows an example configuration where USn_CS is used to automatically enable and disable an external driver.

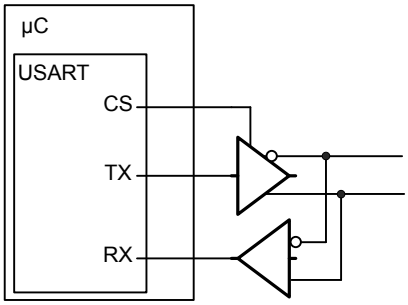


Figure 19.10. USART Half Duplex Communication with External Driver

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).

19.3.2.18 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.

19.3.2.19 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 19.11 USART Transmission of Large Frames on page 719. The first element in the transmit buffer, i.e. element 0 in Figure 19.11 USART Transmission of Large Frames on page 719 is the first element written to the FIFO, or the least significant byte when writing two bytes at a time using USARTn_TXDOUBLE.

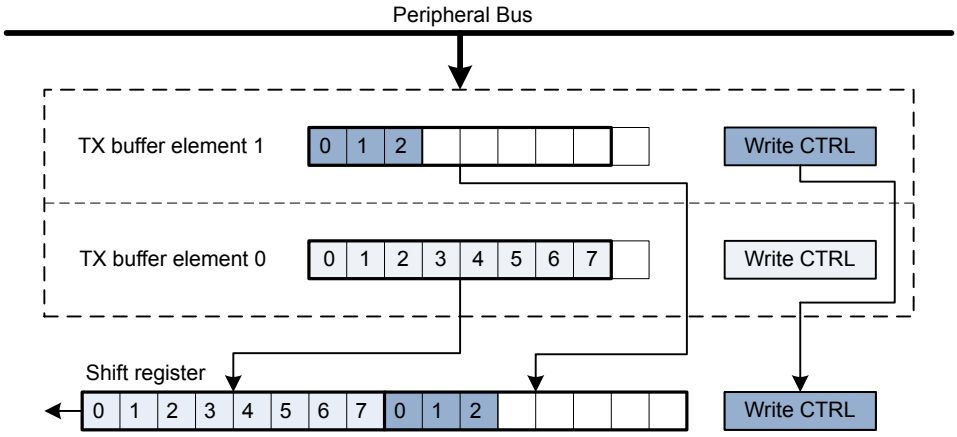


Figure 19.11. USART Transmission of Large Frames

As shown in Figure 19.11 USART Transmission of Large Frames on page 719, 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.

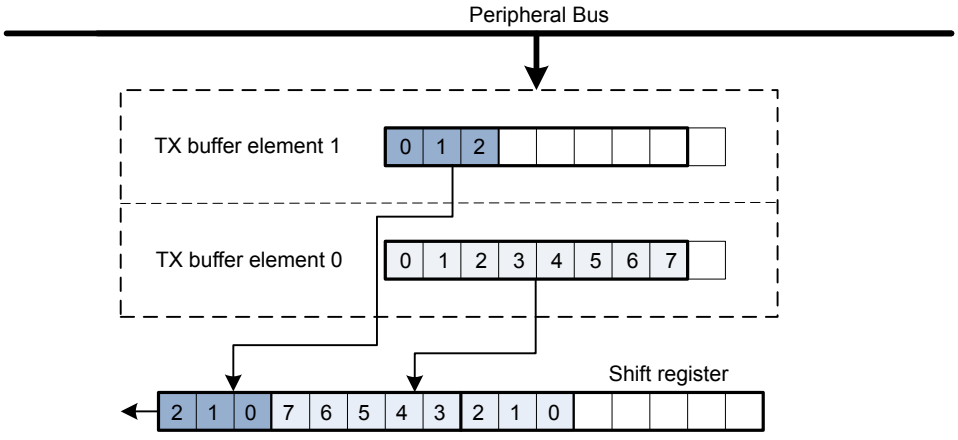


Figure 19.12. USART Transmission of Large Frames, MSBF

Figure 19.12 USART Transmission of Large Frames, MSBF on page 719 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 19.13 USART Reception of Large Frames on page 720. 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.

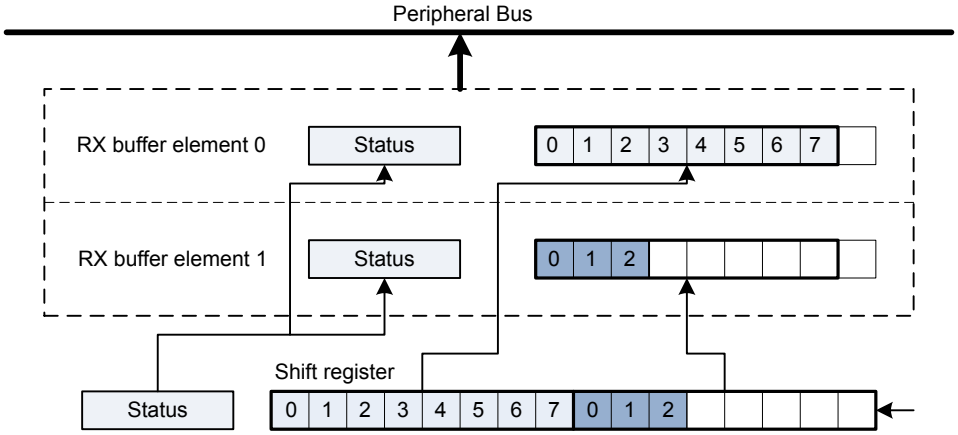


Figure 19.13. USART Reception of Large Frames

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.

19.3.2.20 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.

Basic usage of the multi-processor mode is as follows:

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 USARTn_CTRL is set to identify frames with the 9th bit high as address frames.
2. The master sends a frame containing the address of a slave and with the 9th bit set
3. All slaves receive the address frame and get an interrupt. They can read the address from the receive buffer. The selected slave unblocks the receiver to start receiving data from the master.
4. The master sends data with the 9th bit cleared
5. Only the slave with RX enabled receives the data. When transmission is complete, the slave blocks the receiver and waits for a new address frame.

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.

19.3.2.21 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.

19.3.2.22 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 19.14 USART ISO 7816 Data Frame Without Error on page 722. 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.

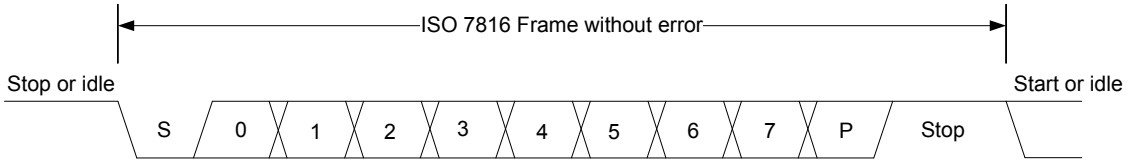


Figure 19.14. USART ISO 7816 Data Frame Without Error

If a parity error is detected by the receiver, it pulls the line I/O line low after half a stop bit, see Figure 19.15 USART ISO 7816 Data Frame With Error on page 722. 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.

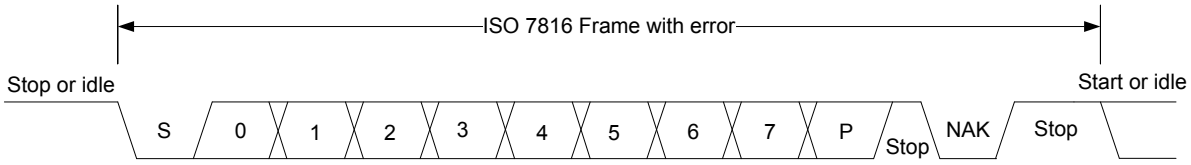


Figure 19.15. USART ISO 7816 Data Frame With Error

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 19.16 USART SmartCard Stop Bit Sampling on page 723. 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.

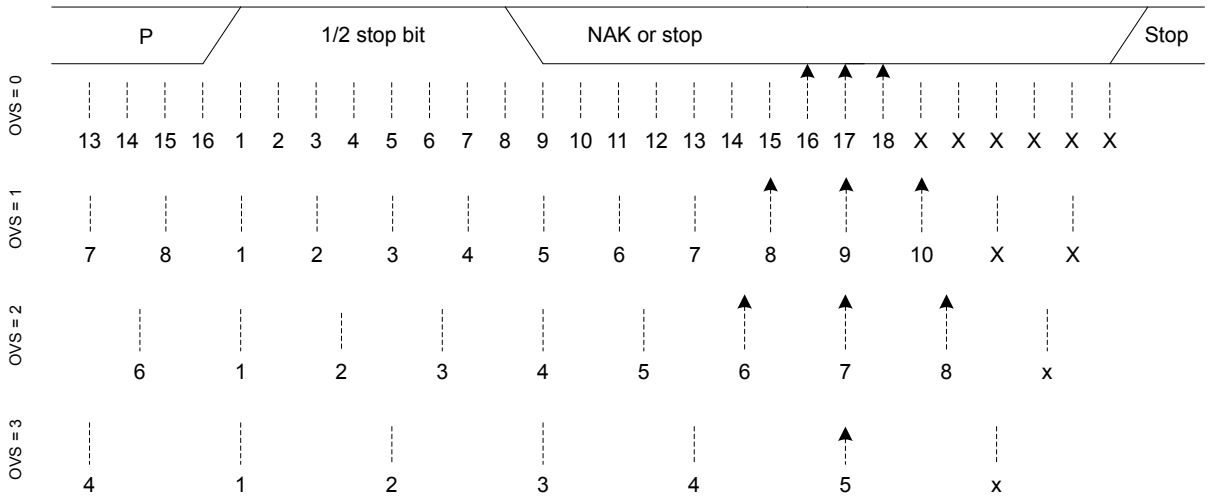


Figure 19.16. USART SmartCard Stop Bit Sampling

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.

19.3.3 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.

19.3.3.1 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.

19.3.3.2 Clock Generation

Note: Not all USART instances are using the same peripheral clock. Normally the USART uses $HFPERCLK_{USARTn}$, however USART2 supports higher frequencies and therefore uses $HFPERBCLK_{USART2}$. This chapter describes the general case and therefore uses $HFPERCLK_{USARTn}$ and $f_{HFPERCLK}$, which should be interpreted as $HFPERBCLK_{USARTn}$ and $f_{HFPERBCLK}$ for USART2. 10.3.1.4 HFPERCLK, HFPERBCLK, HFPERCCLK - High Frequency Peripheral Clocks shows which peripheral uses what peripheral clock.

The bit-rate in synchronous mode is given by Figure 19.17 USART Synchronous Mode Bit Rate on page 724. As in the case of asynchronous operation, the clock division factor have a 15-bit integral part and a 5-bit fractional part.

$$br = f_{HFPERCLK}/(2 \times (1 + USARTn\_CLKDIV/256))$$

Figure 19.17. USART Synchronous Mode Bit Rate

Given a desired baud rate brdesired, the clock divider USARTn_CLKDIV can be calculated using Figure 19.18 USART Synchronous Mode Clock Division Factor on page 724

$$USARTn\_CLKDIV = 256 \times (f_{HFPERCLK}/(2 \times brdesired) - 1)$$

Figure 19.18. 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: $br_{max} = f_{HFPERCLK}/2$
- Slave mode: $br_{max} = f_{HFPERCLK}/8$

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 19.8 USART SPI Modes on page 724. Figure 19.19 USART SPI Timing on page 725 shows the resulting timing of data set-up and sampling relative to the bus clock.

Table 19.8. 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

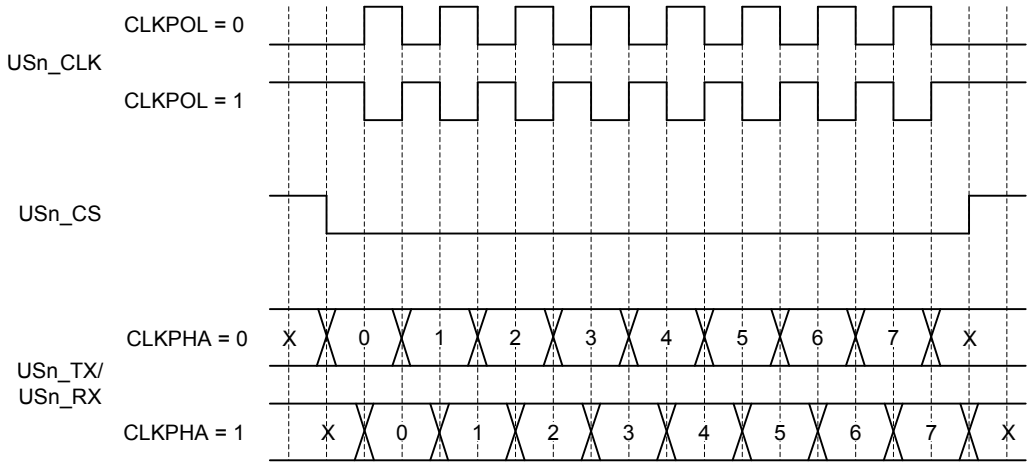


Figure 19.19. USART SPI Timing

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 HFPERCLK 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.

19.3.3.3 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.

19.3.3.4 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.

19.3.3.5 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.

19.3.3.6 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 ROUTE 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.

19.3.3.7 Synchronous Half Duplex Communication

Half duplex communication in synchronous mode is very similar to half duplex communication in asynchronous mode as detailed in 19.3.2.15 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.

19.3.3.8 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.

19.3.3.9 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.

19.3.3.10 Major Modes

The USART supports a set of different I2S formats as shown in Table 19.9 USART I2S Modes on page 728, 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 19.22 USART Left-Justified I2S Waveform on page 729 and Figure 19.23 USART Right-Justified I2S Waveform on page 729. 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 19.9. 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 19.20 USART Standard I2S Waveform on page 728 and Figure 19.21 USART Standard I2S Waveform (Reduced Accuracy) on page 728. 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

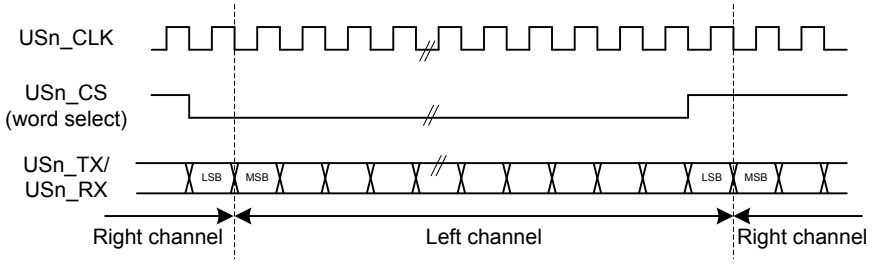


Figure 19.20. USART Standard I2S Waveform

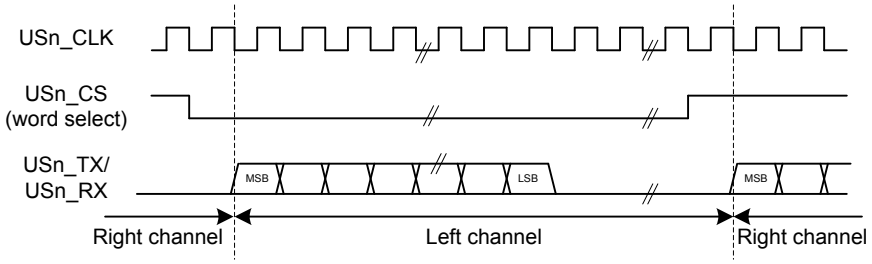


Figure 19.21. USART Standard I2S Waveform (Reduced Accuracy)

A left-justified stream is shown in Figure 19.22 USART Left-Justified I2S Waveform on page 729. 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.

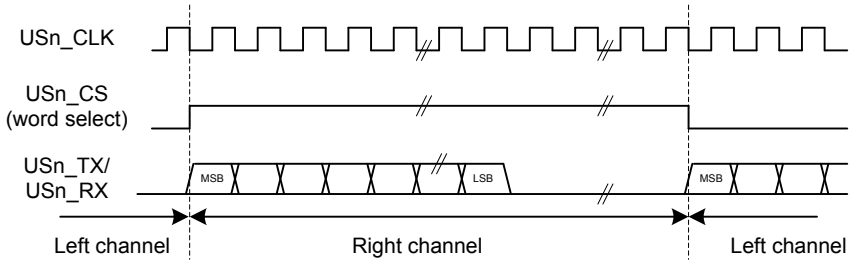


Figure 19.22. USART Left-Justified I2S Waveform

A right-justified stream is shown in Figure 19.23 USART Right-Justified I2S Waveform on page 729. The left and right justified streams are equal when the data-size is equal to the word-width.

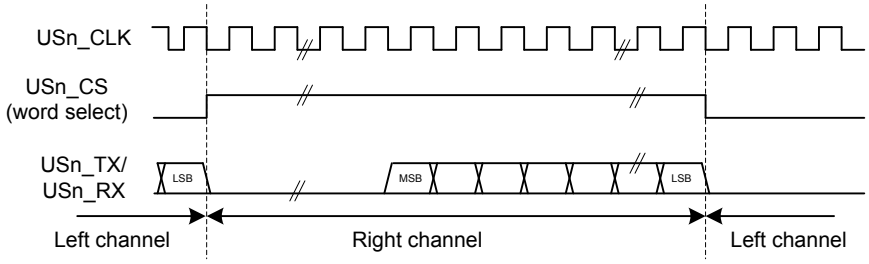


Figure 19.23. USART Right-Justified I2S Waveform

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 19.24 USART Mono I2S Waveform on page 729.

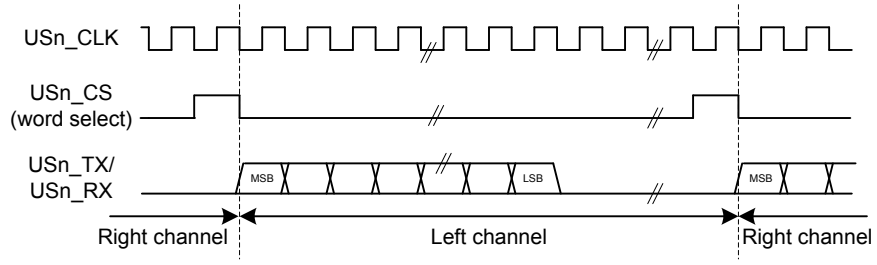


Figure 19.24. USART Mono I2S Waveform

19.3.3.11 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 RXFULL-RIGHT 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.

19.3.4 Hardware Flow Control

Hardware flow control can be used to hold off the link partner's transmission until RX buffer space is available. Use RTSPEN and CTSPEN in USARTn_ROUTEPEN to allocate the hardware flow control to GPIOs. 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.

19.3.5 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 deas-sert 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.

19.3.6 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 TSEL in USARTn_TRIGCTRL. 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 TSEL 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

19.3.7 PRS RX Input

The USART can be configured to receive data directly from a PRS channel by setting RXPRS in USARTn_INPUT. The PRS channel used is selected using RXPRSSEL in USARTn_INPUT. This way, for example, a differential RX signal can be input to the ACMP and the output routed via PRS to the USART.

19.3.8 PRS CLK Input

The USART can be configured to receive clock directly from a PRS channel by setting CLKPRS in USARTn_INPUT. The PRS channel used is selected using CLKPRSSEL in USARTn_INPUT. This is useful in synchronous slave mode and can together with RX PRS input be used to input data from PRS.

19.3.9 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.

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.

19.3.10 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 19.25 USART Timer Block Diagram on page 733.

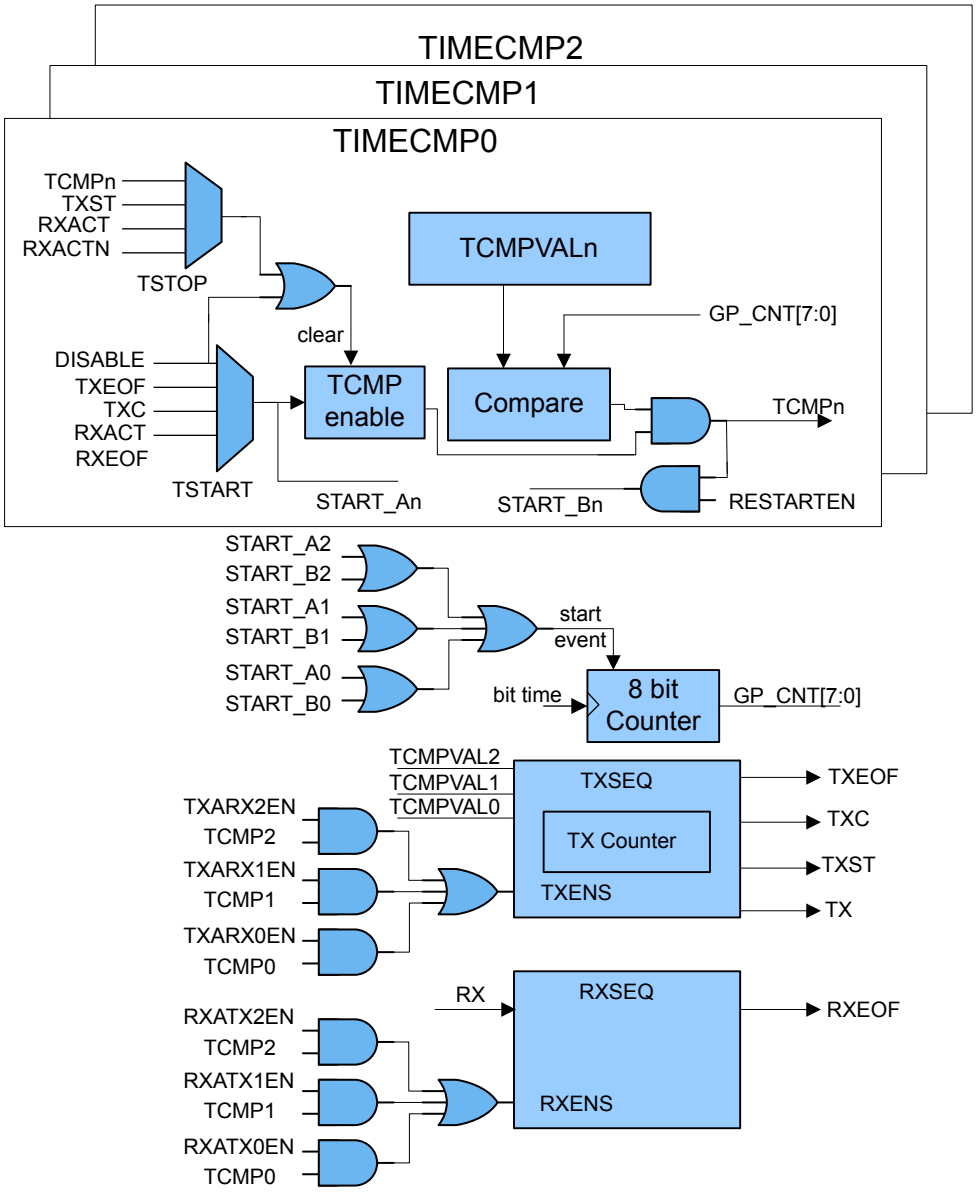


Figure 19.25. USART Timer Block Diagram

The following sections will go into more details on programming the various usage cases.

Table 19.10. 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; TIMERRESTARTED in USARTn_STATUS; 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 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.

19.3.10.1 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 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.

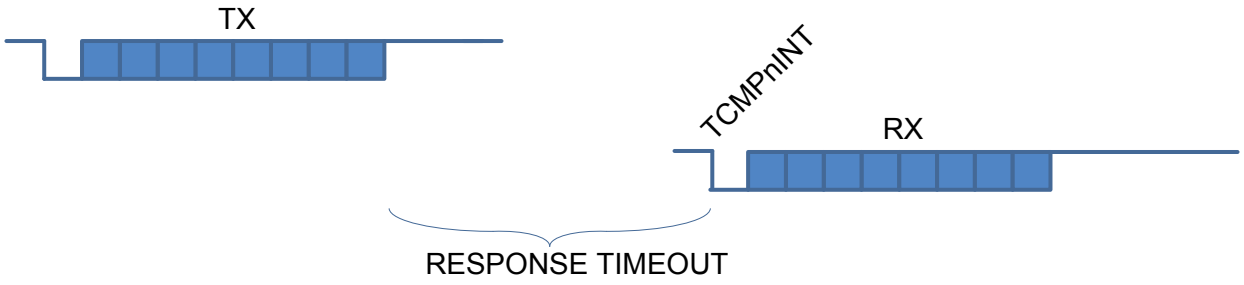


Figure 19.26. USART Response Timeout

19.3.10.2 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 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.

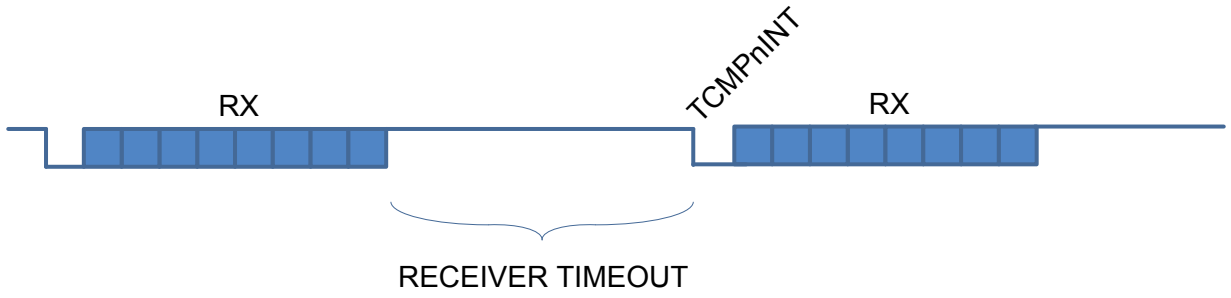


Figure 19.27. USART RX Timeout

19.3.10.3 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 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.

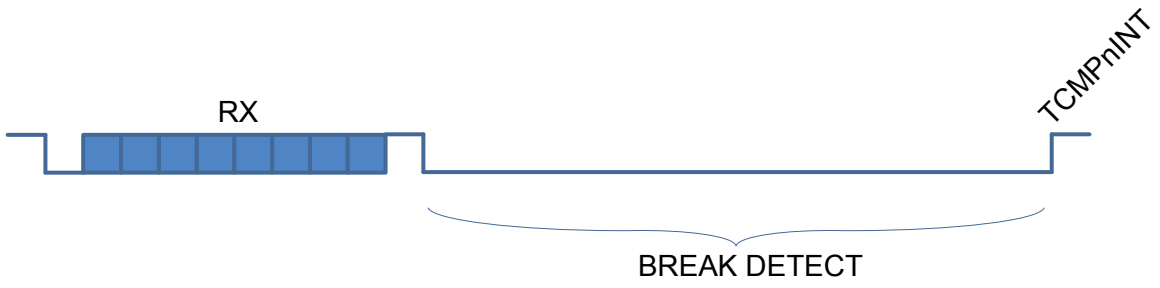


Figure 19.28. USART Break Detection

19.3.10.4 TX Start Delay

Some applications may require a delay before the start of transmission. This example in Figure 19.29 USART TXSEQ Timing on page 736 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 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 19.29. USART TXSEQ Timing

19.3.10.5 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 for details on setting up this example. For this example in Figure 19.29 USART TXSEQ Timing on page 736 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.

19.3.10.6 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 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 19.29 USART TXSEQ Timing on page 736.

19.3.10.7 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 for details on setting up this example. TXARX1EN in USARTn_TRIGCTRL tells the TX sequencer to trigger after RX EOF plus tcmp1val baud times.

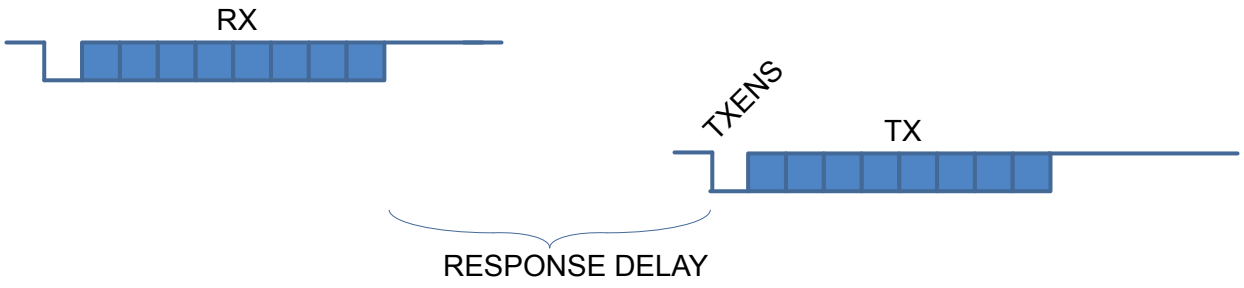


Figure 19.30. USART Response Delay

19.3.10.8 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 for details on setting up this example.

19.3.10.9 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 19.10 USART Application Settings for USARTn_TIMING and USARTn_TIMECMPn on page 733 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.

19.3.10.10 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.

19.3.11 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.

19.3.12 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 19.31 USART Example RZI Signal for a given Asynchronous USART Frame on page 738.

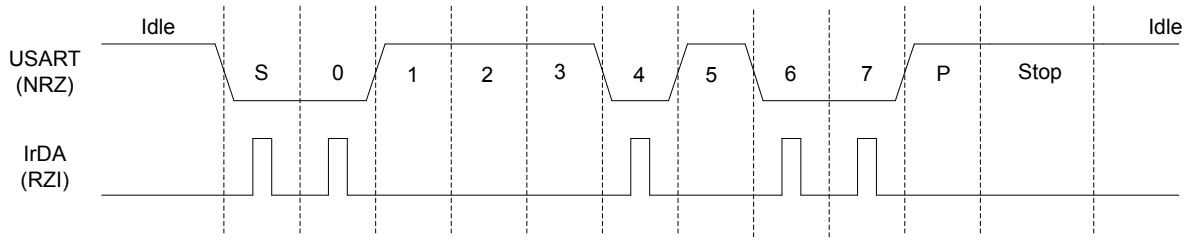


Figure 19.31. USART Example RZI Signal for a given Asynchronous USART Frame

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 19.11 USART IrDA Pulse Widths on page 738.

Table 19.11. 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.

The IrDA module can also modulate a signal from the PRS system, and transmit a modulated signal to the PRS system. To use a PRS channel as transmitter source instead of the USART, set IRPRSEN in USARTn_IRCTRL high. The channel is selected by configuring IRPRSSEL in USARTn_IRCTRL.

19.4 Register Map

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

Offset	Name	Type	Description
0x000	USARTn_CTRL	RW	Control Register
0x004	USARTn_FRAME	RW	USART Frame Format Register
0x008	USARTn_TRIGCTRL	RW	USART Trigger Control Register
0x00C	USARTn_CMD	W1	Command Register
0x010	USARTn_STATUS	R	USART Status Register
0x014	USARTn_CLKDIV	RWH	Clock Control Register
0x018	USARTn_RXDATAX	R(a)	RX Buffer Data Extended Register
0x01C	USARTn_RXDATA	R(a)	RX Buffer Data Register
0x020	USARTn_RXDOUBLEX	R(a)	RX Buffer Double Data Extended Register
0x024	USARTn_RXDOUBLE	R(a)	RX FIFO Double Data Register
0x028	USARTn_RXDATAXP	R	RX Buffer Data Extended Peek Register
0x02C	USARTn_RXDOUBLEXP	R	RX Buffer Double Data Extended Peek Register
0x030	USARTn_TXDATAX	W	TX Buffer Data Extended Register
0x034	USARTn_TXDATA	W	TX Buffer Data Register
0x038	USARTn_TXDOUBLEX	W	TX Buffer Double Data Extended Register
0x03C	USARTn_TXDOUBLE	W	TX Buffer Double Data Register
0x040	USARTn_IF	R	Interrupt Flag Register
0x044	USARTn_IFS	W1	Interrupt Flag Set Register
0x048	USARTn_IFC	(R)W1	Interrupt Flag Clear Register
0x04C	USARTn_IEN	RW	Interrupt Enable Register
0x050	USARTn_IRCTRL	RW	IrDA Control Register
0x058	USARTn_INPUT	RW	USART Input Register
0x05C	USARTn_I2SCTRL	RW	I2S Control Register
0x060	USARTn_TIMING	RW	Timing Register
0x064	USARTn_CTRLX	RW	Control Register Extended
0x068	USARTn_TIMECMP0	RW	Used to Generate Interrupts and Various Delays
0x06C	USARTn_TIMECMP1	RW	Used to Generate Interrupts and Various Delays
0x070	USARTn_TIMECMP2	RW	Used to Generate Interrupts and Various Delays
0x074	USARTn_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x078	USARTn_ROUTELOC0	RW	I/O Routing Location Register
0x07C	USARTn_ROUTELOC1	RW	I/O Routing Location Register

19.5 Register Description

19.5.1 USARTn_CTRL - 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	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0x0		0	0	0	0	0
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	0	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	0	RW	**Majority Vote Disable**
	Disable majority vote for 16x, 8x and 6x oversampling modes.			
29	AUTOTX	0	RW	**Always Transmit When RX Not Full**
	Transmits as long as RX is not full. If TX is empty, underflows are generated.			
28	BYTESWAP	0	RW	**Byteswap in Double Accesses**
	Set to switch the order of the bytes in double accesses.			

Value	Description
0	Normal byte order
1	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	0	RW	**Synchronous Slave Setup Early**
	Setup data on sample edge in synchronous slave mode to improve MOSI setup time			
24	ERRSTX	0	RW	**Disable TX on Error**
	When set, the transmitter is disabled on framing and parity errors (asynchronous mode only) in the receiver.			

Value	Description
0	Received framing and parity errors have no effect on transmitter
1	Received framing and parity errors disable the transmitter

23	ERRSRX	0	RW	**Disable RX on Error**
	When set, the receiver is disabled on framing and parity errors (asynchronous mode only).			

Value	Description
0	Framing and parity errors have no effect on receiver

Bit	Name	Reset	Access	Description
	1			Framing and parity errors disable the receiver
22	ERRSDMA	0	RW	**Halt DMA on Error**

When set, DMA requests will be cleared on framing and parity errors (asynchronous mode only).

Value	Description
0	Framing and parity errors have no effect on DMA requests from the USART
1	DMA requests from the USART are blocked while the PERR or FERR interrupt flags are set

21	BIT8DV	0	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.

20	SKIPPERRF	0	RW	**Skip Parity Error Frames**

When set, the receiver discards frames with parity errors (asynchronous mode only). The PERR interrupt flag is still set.

19	SCRETRANS	0	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.

18	SCMODE	0	RW	**SmartCard Mode**

Use this bit to enable or disable SmartCard mode.

17	AUTOTRI	0	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	Description
0	The output on U(S)n_TX when the transmitter is idle is defined by TXINV
1	U(S)n_TX is tristated whenever the transmitter is idle

16	AUTOCS	0	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.

15	CSINV	0	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	Description
0	Chip select is active low
1	Chip select is active high

14	TXINV	0	RW	**Transmitter Output Invert**

The output from the USART transmitter can optionally be inverted by setting this bit.

Value	Description
0	Output from the transmitter is passed unchanged to U(S)n_TX

Bit	Name	Reset	Access	Description
	1			Output from the transmitter is inverted before it is passed to U(S)n_TX
13	RXINV	0	RW	**Receiver Input Invert**

Setting this bit will invert the input to the USART receiver.

Value	Description
0	Input is passed directly to the receiver
1	Input is inverted before it is passed to the receiver

12	TXBIL	0	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.

11	CSMA	0	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

10	MSBF	0	RW	**Most Significant Bit First**

Decides whether data is sent with the least significant bit first, or the most significant bit first.

Value	Description
0	Data is sent with the least significant bit first
1	Data is sent with the most significant bit first

9	CLKPHA	0	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

8	CLKPOL	0	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	0	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	0	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			Description
	0			The 9th bit of incoming frames has no special function
	1			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	0	RW	**Collision Check Enable**
	Enables collision checking on data when operating in half duplex modus.			
	Value			Description
	0			Collision check is disabled
	1			Collision check is enabled. The receiver must be enabled for the check to be performed
1	LOOPBK	0	RW	**Loopback Enable**
	Allows the receiver to be connected directly to the USART transmitter for loopback and half duplex communication.			
	Value			Description
	0			The receiver is connected to and receives data from U(S)n_RX
	1			The receiver is connected to and receives data from U(S)n_TX
0	SYNC	0	RW	**USART Synchronous Mode**
	Determines whether the USART is operating in asynchronous or synchronous mode.			
	Value			Description
	0			The USART operates in asynchronous mode

Bit	Name	Reset	Access	Description
	1			The USART operates in synchronous mode

19.5.2 USARTn_FRAME - USART Frame Format 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																			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

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	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.

Bit	Name	Reset	Access	Description
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

19.5.3 USARTn_TRIGCTRL - USART Trigger 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							0	0	0	0	0	0	0	0	0				
Access													RW							RW	RW	RW	RW	RW	RW	RW	RW	RW				
Name													TSEL							RXATX2EN	RXATX1EN	RXATX0EN	TXARX2EN	TXARX1EN	TXARX0EN	AUTOTXTEN	TXTEN	RXTEN				

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	TSEL	0x0	RW	**Trigger PRS Channel Select**

Select USART PRS trigger channel. The PRS signal can enable RX and/or TX, depending on the setting of RXTEN and TXTEN.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected
8	PRSCH8	PRS Channel 8 selected
9	PRSCH9	PRS Channel 9 selected
10	PRSCH10	PRS Channel 10 selected
11	PRSCH11	PRS Channel 11 selected
12	PRSCH12	PRS Channel 12 selected
13	PRSCH13	PRS Channel 13 selected
14	PRSCH14	PRS Channel 14 selected
15	PRSCH15	PRS Channel 15 selected

15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	RXATX2EN	0	RW	**Enable Receive Trigger After TX End of Frame Plus TCMPVAL2 Baud-times**

When set, a TX end of frame will trigger the receiver after a TCMPVAL2 baud-time delay

Bit	Name	Reset	Access	Description
11	RXATX1EN	0	RW	**Enable Receive Trigger After TX End of Frame Plus TCMPVAL1 Baud-times**
	When set, a TX end of frame will trigger the receiver after a TCMPVAL1 baud-time delay			
10	RXATX0EN	0	RW	**Enable Receive Trigger After TX End of Frame Plus TCMPVAL0 Baud-times**
	When set, a TX end of frame will trigger the receiver after a TCMPVAL0 baud-time delay			
9	TXARX2EN	0	RW	**Enable Transmit Trigger After RX End of Frame Plus TCMP2VAL**
	When set, an RX end of frame will trigger the transmitter after TCMP2VAL bit times to force a minimum response delay			
8	TXARX1EN	0	RW	**Enable Transmit Trigger After RX End of Frame Plus TCMP1VAL**
	When set, an RX end of frame will trigger the transmitter after TCMP1VAL bit times to force a minimum response delay			
7	TXARX0EN	0	RW	**Enable Transmit Trigger After RX End of Frame Plus TCMP0VAL**
	When set, an RX end of frame will trigger the transmitter after TCMP0VAL bit times to force a minimum response delay			
6	AUTOTXTEN	0	RW	**AUTOTX Trigger Enable**
	When set, AUTOTX is enabled as long as the PRS channel selected by TSEL has a high value			
5	TXTEN	0	RW	**Transmit Trigger Enable**
	When set, the PRS channel selected by TSEL sets TXEN, enabling the transmitter on positive trigger edges.			
4	RXTEN	0	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*		

19.5.4 USARTn_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																					0	0	0	0	0	0	0	0	0	0	0	0
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
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	0	W1	**Clear RX**
	Set to clear receive buffer and the RX shift register.			
10	CLEARTX	0	W1	**Clear TX**
	Set to clear transmit buffer and the TX shift register.			
9	TXTRIDIS	0	W1	**Transmitter Tristate Disable**
	Disables tristating of the transmitter output.			
8	TXTRIEN	0	W1	**Transmitter Tristate Enable**
	Tristates the transmitter output.			
7	RXBLOCKDIS	0	W1	**Receiver Block Disable**
	Set to clear RXBLOCK, resulting in all incoming frames being loaded into the receive buffer.			
6	RXBLOCKEN	0	W1	**Receiver Block Enable**
	Set to set RXBLOCK, resulting in all incoming frames being discarded.			
5	MASTERDIS	0	W1	**Master Disable**
	Set to disable master mode, clearing the MASTER status bit and putting the USART in slave mode.			
4	MASTEREN	0	W1	**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	0	W1	**Transmitter Disable**
	Set to disable transmission.			
2	TXEN	0	W1	**Transmitter Enable**
	Set to enable data transmission.			
1	RXDIS	0	W1	**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	0	W1	**Receiver Enable**
	Set to activate data reception on U(S)n_RX.			

19.5.5 USARTn_STATUS - USART 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			0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
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	0	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	1	R	**TX Idle**
	Set when TX idle			
12	RXFULLRIGHT	0	R	**RX Full of Right Data**
	When set, the entire RX buffer contains right data. Only used in I2S mode			
11	RXDATAVRIGHT	0	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	0	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	0	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	0	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	0	R	**RX Data Valid**
	Set when data is available in the receive buffer. Cleared when the receive buffer is empty.			
6	TXBL	1	R	**TX Buffer Level**
	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.			

Bit	Name	Reset	Access	Description
5	TXC	0	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	0	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	0	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	0	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	0	R	**Transmitter Enable Status**
	Set when the transmitter is enabled.			
0	RXENS	0	R	**Receiver Enable Status**
	Set when the receiver is enabled.			

19.5.6 USARTn_CLKDIV - Clock Control 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	0									0x00000																						
Access	RW									RWH																						
Name	AUTOBAUDEN									DIV																						

Bit	Name	Reset	Access	Description
31	AUTOBAUDEN	0	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	0x00000	RWH	**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*		

19.5.7 USARTn_RXDATAX - RX Buffer Data Extended Register (Actionable Reads)

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																	0	0						0x000								
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	0	R	**Data Framing Error**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERR	0	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	0x000	R	**RX Data**
	Use this register to access data read from the USART. Buffer is cleared on read access.			

19.5.8 USARTn_RXDATA - RX Buffer Data Register (Actionable Reads)

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																												0x00				
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	0x00	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.			

19.5.9 USARTn_RXDOUBLEX - RX Buffer Double Data Extended Register (Actionable Reads)

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	0	0						0x000									0	0						0x000								
Access	R	R						R									R	R						R								
Name	FERR1	PERR1						RXDATA1									FERR0	PERR0						RXDATA0								

Bit	Name	Reset	Access	Description
31	FERR1	0	R	**Data Framing Error 1**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
30	PERR1	0	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	0x000	R	**RX Data 1**
	Second frame read from buffer.			
15	FERR0	0	R	**Data Framing Error 0**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERR0	0	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	0x000	R	**RX Data 0**
	First frame read from buffer.			

19.5.10 USARTn_RXDOUBLE - RX FIFO Double Data Register (Actionable Reads)

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																	0x00								0x00							
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	0x00	R	**RX Data 1**
	Second frame read from buffer.			
7:0	RXDATA0	0x00	R	**RX Data 0**
	First frame read from buffer.			

19.5.11 USARTn_RXDATAXP - RX Buffer Data Extended Peek 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																	0	0							0x000							
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	0	R	**Data Framing Error Peek**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERRP	0	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	0x000	R	**RX Data Peek**
	Use this register to access data read from the USART.			

19.5.12 USARTn_RXDOUBLEXP - RX Buffer Double Data Extended 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	0	0										0x000					0	0										0x000				
Access	R	R										R					R	R										R				
Name	FERRP1	PERRP1										RXDATAP1					FERRP0	PERRP0										RXDATAP0				

Bit	Name	Reset	Access	Description
31	FERRP1	0	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	0	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	0x000	R	**RX Data 1 Peek**
	Second frame read from FIFO.			
15	FERRP0	0	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	0	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	0x000	R	**RX Data 0 Peek**
	First frame read from FIFO.			

19.5.13 USARTn_TXDATAX - TX Buffer Data Extended 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																	0	0	0	0	0			0x000								
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	0	W	**Enable RX After Transmission**
	Set to enable reception after transmission.			
14	TXDISAT	0	W	**Clear TXEN After Transmission**
	Set to disable transmitter and release data bus directly after transmission.			
13	TXBREAK	0	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	0	W	**Set TXTRI After Transmission**
	Set to tristate transmitter by setting TXTRI after transmission.			
11	UBRXAT	0	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	0x000	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.			

19.5.14 USARTn_TXDATA - TX 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																									0x00							
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	0x00	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.			

19.5.15 USARTn_TXDOUBLEX - TX Buffer Double 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	0	0	0	0	0			0x000									0	0	0	0	0			0x000								
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	0	W	**Enable RX After Transmission**
	Set to enable reception after transmission.			
30	TXDISAT1	0	W	**Clear TXEN After Transmission**
	Set to disable transmitter and release data bus directly after transmission.			
29	TXBREAK1	0	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	0	W	**Set TXTRI After Transmission**
	Set to tristate transmitter by setting TXTRI after transmission.			
27	UBRXAT1	0	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	0x000	W	**TX Data**
	Second frame to write to FIFO.			
15	RXENAT0	0	W	**Enable RX After Transmission**
	Set to enable reception after transmission.			
14	TXDISAT0	0	W	**Clear TXEN After Transmission**
	Set to disable transmitter and release data bus directly after transmission.			
13	TXBREAK0	0	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	0	W	**Set TXTRI After Transmission**
	Set to tristate transmitter by setting TXTRI after transmission.			
11	UBRXAT0	0	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	0x000	W	**TX Data**
	First frame to write to buffer.			

19.5.16 USARTn_TXDOUBLE - TX Buffer Double 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																	0x00								0x00							
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	0x00	W	**TX Data**
	Second frame to write to buffer.			
7:0	TXDATA0	0x00	W	**TX Data**
	First frame to write to buffer.			

19.5.17 USARTn_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																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Access																R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
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	0	R	**Timer Comparator 2 Interrupt Flag**
	Set when the timer reaches the comparator 2 value, TCMP2.			
15	TCMP1	0	R	**Timer Comparator 1 Interrupt Flag**
	Set when the timer reaches the comparator 1 value, TCMP1.			
14	TCMP0	0	R	**Timer Comparator 0 Interrupt Flag**
	Set when the Timer reaches the comparator 0 value, TCMP0.			
13	TXIDLE	0	R	**TX Idle Interrupt Flag**
	Set when TX goes idle. At this point, transmission has ended			
12	CCF	0	R	**Collision Check Fail Interrupt Flag**
	Set when a collision check notices an error in the transmitted data.			
11	SSM	0	R	**Slave-Select in Master Mode Interrupt Flag**
	Set when the device is selected as a slave when in master mode.			
10	MPAF	0	R	**Multi-Processor Address Frame Interrupt Flag**
	Set when a multi-processor address frame is detected.			
9	FERR	0	R	**Framing Error Interrupt Flag**
	Set when a frame with a framing error is received while RXBLOCK is cleared.			
8	PERR	0	R	**Parity Error Interrupt Flag**
	Set when a frame with a parity error (asynchronous mode only) is received while RXBLOCK is cleared.			
7	TXUF	0	R	**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	0	R	**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	0	R	**RX Underflow Interrupt Flag**
	Set when trying to read from the receive buffer when it is empty.			
4	RXOF	0	R	**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	0	R	**RX Buffer Full Interrupt Flag**
	Set when the receive buffer becomes full.			
2	RXDATAV	0	R	**RX Data Valid Interrupt Flag**
	Set when data becomes available in the receive buffer.			
1	TXBL	1	R	**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	0	R	**TX Complete Interrupt Flag**
	This interrupt is set after a transmission when both the TX buffer and shift register are empty.			

19.5.18 USARTn_IFS - Interrupt Flag Set 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																0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
Access																W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1			W1
Name																TCMP2	TCMP1	TCMP0	TXIDLE	CCF	SSM	MPAF	FERR	PERR	TXUF	TXOF	RXUF	RXOF	RXFULL			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	0	W1	**Set TCMP2 Interrupt Flag**
	Write 1 to set the TCMP2 interrupt flag			
15	TCMP1	0	W1	**Set TCMP1 Interrupt Flag**
	Write 1 to set the TCMP1 interrupt flag			
14	TCMP0	0	W1	**Set TCMP0 Interrupt Flag**
	Write 1 to set the TCMP0 interrupt flag			
13	TXIDLE	0	W1	**Set TXIDLE Interrupt Flag**
	Write 1 to set the TXIDLE interrupt flag			
12	CCF	0	W1	**Set CCF Interrupt Flag**
	Write 1 to set the CCF interrupt flag			
11	SSM	0	W1	**Set SSM Interrupt Flag**
	Write 1 to set the SSM interrupt flag			
10	MPAF	0	W1	**Set MPAF Interrupt Flag**
	Write 1 to set the MPAF interrupt flag			
9	FERR	0	W1	**Set FERR Interrupt Flag**
	Write 1 to set the FERR interrupt flag			
8	PERR	0	W1	**Set PERR Interrupt Flag**
	Write 1 to set the PERR interrupt flag			
7	TXUF	0	W1	**Set TXUF Interrupt Flag**
	Write 1 to set the TXUF interrupt flag			
6	TXOF	0	W1	**Set TXOF Interrupt Flag**
	Write 1 to set the TXOF interrupt flag			
5	RXUF	0	W1	**Set RXUF Interrupt Flag**
	Write 1 to set the RXUF interrupt flag			
4	RXOF	0	W1	**Set RXOF Interrupt Flag**
	Write 1 to set the RXOF interrupt flag			
3	RXFULL	0	W1	**Set RXFULL Interrupt Flag**
	Write 1 to set the RXFULL interrupt flag			

Bit	Name	Reset	Access	Description
2:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	TXC	0	W1	**Set TXC Interrupt Flag**
	Write 1 to set the TXC interrupt flag			

19.5.19 USARTn_IFC - Interrupt Flag Clear 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																0	0	0	0	0	0	0	0	0	0	0	0	0	0			0
Access																(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1			(R)W1
Name																TCMP2	TCMP1	TCMP0	TXIDLE	CCF	SSM	MPAF	FERR	PERR	TXUF	TXOF	RXUF	RXOF	RXFULL			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	0	(R)W1	**Clear TCMP2 Interrupt Flag**
	Write 1 to clear the TCMP2 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
15	TCMP1	0	(R)W1	**Clear TCMP1 Interrupt Flag**
	Write 1 to clear the TCMP1 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
14	TCMP0	0	(R)W1	**Clear TCMP0 Interrupt Flag**
	Write 1 to clear the TCMP0 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
13	TXIDLE	0	(R)W1	**Clear TXIDLE Interrupt Flag**
	Write 1 to clear the TXIDLE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
12	CCF	0	(R)W1	**Clear CCF Interrupt Flag**
	Write 1 to clear the CCF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
11	SSM	0	(R)W1	**Clear SSM Interrupt Flag**
	Write 1 to clear the SSM interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
10	MPAF	0	(R)W1	**Clear MPAF Interrupt Flag**
	Write 1 to clear the MPAF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	FERR	0	(R)W1	**Clear FERR Interrupt Flag**
	Write 1 to clear the FERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	PERR	0	(R)W1	**Clear PERR Interrupt Flag**
	Write 1 to clear the PERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	TXUF	0	(R)W1	**Clear TXUF Interrupt Flag**
	Write 1 to clear the TXUF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
6	TXOF	0	(R)W1	**Clear TXOF Interrupt Flag**
	Write 1 to clear the TXOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
5	RXUF	0	(R)W1	**Clear RXUF Interrupt Flag**
	Write 1 to clear the RXUF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
4	RXOF	0	(R)W1	**Clear RXOF Interrupt Flag**
	Write 1 to clear the RXOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	RXFULL	0	(R)W1	**Clear RXFULL Interrupt Flag**
	Write 1 to clear the RXFULL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	TXC	0	(R)W1	**Clear TXC Interrupt Flag**
	Write 1 to clear the TXC interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

19.5.20 USARTn_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																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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	0	RW	**TCMP2 Interrupt Enable**
	Enable/disable the TCMP2 interrupt			
15	TCMP1	0	RW	**TCMP1 Interrupt Enable**
	Enable/disable the TCMP1 interrupt			
14	TCMP0	0	RW	**TCMP0 Interrupt Enable**
	Enable/disable the TCMP0 interrupt			
13	TXIDLE	0	RW	**TXIDLE Interrupt Enable**
	Enable/disable the TXIDLE interrupt			
12	CCF	0	RW	**CCF Interrupt Enable**
	Enable/disable the CCF interrupt			
11	SSM	0	RW	**SSM Interrupt Enable**
	Enable/disable the SSM interrupt			
10	MPAF	0	RW	**MPAF Interrupt Enable**
	Enable/disable the MPAF interrupt			
9	FERR	0	RW	**FERR Interrupt Enable**
	Enable/disable the FERR interrupt			
8	PERR	0	RW	**PERR Interrupt Enable**
	Enable/disable the PERR interrupt			
7	TXUF	0	RW	**TXUF Interrupt Enable**
	Enable/disable the TXUF interrupt			
6	TXOF	0	RW	**TXOF Interrupt Enable**
	Enable/disable the TXOF interrupt			
5	RXUF	0	RW	**RXUF Interrupt Enable**
	Enable/disable the RXUF interrupt			
4	RXOF	0	RW	**RXOF Interrupt Enable**
	Enable/disable the RXOF interrupt			

Bit	Name	Reset	Access	Description
3	RXFULL	0	RW	**RXFULL Interrupt Enable**
	Enable/disable the RXFULL interrupt			
2	RXDATAV	0	RW	**RXDATAV Interrupt Enable**
	Enable/disable the RXDATAV interrupt			
1	TXBL	0	RW	**TXBL Interrupt Enable**
	Enable/disable the TXBL interrupt			
0	TXC	0	RW	**TXC Interrupt Enable**
	Enable/disable the TXC interrupt			

19.5.21 USARTn_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				0				0	0x0		0
Access																					RW				RW				RW	RW		RW
Name																					IRPRSSEL				IRPRSEN				IRFILT	IRPW		IREN

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*		
11:8	IRPRSSEL	0x0	RW	**IrDA PRS Channel Select**

A PRS can be used as input to the pulse modulator instead of TX. This value selects the channel to use.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected
8	PRSCH8	PRS Channel 8 selected
9	PRSCH9	PRS Channel 9 selected
10	PRSCH10	PRS Channel 10 selected
11	PRSCH11	PRS Channel 11 selected
12	PRSCH12	PRS Channel 12 selected
13	PRSCH13	PRS Channel 13 selected
14	PRSCH14	PRS Channel 14 selected
15	PRSCH15	PRS Channel 15 selected

7	IRPRSEN	0	RW	**IrDA PRS Channel Enable**

Enable the PRS channel selected by IRPRSSEL as input to IrDA module instead of TX.

6:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	IRFILT	0	RW	**IrDA RX Filter**

Set to enable filter on IrDA demodulator.

Value	Description

Bit	Name	Reset	Access	Description
	0			No filter enabled
	1			Filter enabled. IrDA pulse must be high for at least 4 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	0	RW	**Enable IrDA Module**
	Enable IrDA module and rout USART signals through it.			

19.5.22 USARTn_INPUT - USART Input 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																	0				0x0				0				0x0			
Access																	RW				RW				RW				RW			
Name																	CLKPRS				CLKPRSSEL				RXPRS				RXPRSSEL			

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	CLKPRS	0	RW	**PRS CLK Enable**
	When set, the PRS channel selected as input to CLK.			
14:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	CLKPRSSEL	0x0	RW	**CLK PRS Channel Select**
	Select PRS channel as input to CLK.			

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected
8	PRSCH8	PRS Channel 8 selected
9	PRSCH9	PRS Channel 9 selected
10	PRSCH10	PRS Channel 10 selected
11	PRSCH11	PRS Channel 11 selected
12	PRSCH12	PRS Channel 12 selected
13	PRSCH13	PRS Channel 13 selected
14	PRSCH14	PRS Channel 14 selected
15	PRSCH15	PRS Channel 15 selected

7	RXPRS	0	RW	**PRS RX Enable**
	When set, the PRS channel selected as input to RX.			

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*		
3:0	RXPRSSEL	0x0	RW	**RX PRS Channel Select**

Select PRS channel as input to RX.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected
8	PRSCH8	PRS Channel 8 selected
9	PRSCH9	PRS Channel 9 selected
10	PRSCH10	PRS Channel 10 selected
11	PRSCH11	PRS Channel 11 selected
12	PRSCH12	PRS Channel 12 selected
13	PRSCH13	PRS Channel 13 selected
14	PRSCH14	PRS Channel 14 selected
15	PRSCH15	PRS Channel 15 selected

19.5.23 USARTn_I2SCTRL - I2S Control 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						0	0	0	0	0
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	0	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	0	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	0	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	0	RW	**Stero or Mono**

Switch between stereo and mono mode. Set for mono

Bit	Name	Reset	Access	Description
0	EN	0	RW	**Enable I2S Mode**
	Set the U(S)ART in I2S mode.			

19.5.24 USARTn_TIMING - Timing 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																		
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*		

19.5.25 USARTn_CTRLX - Control Register Extended

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																													0	0	0	0
Access																													RW	RW	RW	RW
Name																													RTSINV	CTSEN	CTSINV	DBGHALT

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	RTSINV	0	RW	**RTS Pin Inversion**

When set, the RTS pin polarity is inverted.

Value	Description
0	The USn_RTS pin is low true
1	The USn_RTS pin is high true

Bit	Name	Reset	Access	Description
2	CTSEN	0	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	Description
0	Ingore CTS
1	Stop transmitting when CTS is negated

Bit	Name	Reset	Access	Description
1	CTSINV	0	RW	**CTS Pin Inversion**

When set, the CTS pin polarity is inverted.

Value	Description
0	The USn_CTS pin is low true
1	The USn_CTS pin is high true

Bit	Name	Reset	Access	Description
0	DBGHALT	0	RW	**Debug Halt**

.

Value	Description
0	Continue to transmit until TX buffer is empty
1	Complete the transmission in the shift register and then halt transmission; also 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.

19.5.26 USARTn_TIMECMP0 - Used to Generate Interrupts and Various Delays

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								0		0x0				0x0											0x00							
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	0	RW	**Restart Timer on TCMP0**

Each TCMP0 event will reset and restart the timer

Value	Description
0	Disable the timer restarting on TCMP0
1	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 the start of transmission
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 Active (default: low)
4	RXEOF	Comparator 0 and timer are started at RX end of frame

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	TCMPVAL	0x00	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.

19.5.27 USARTn_TIMECMP1 - Used to Generate Interrupts and Various Delays

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								0		0x0				0x0											0x00							
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	0	RW	**Restart Timer on TCMP1**

Each TCMP1 event will reset and restart the timer

Value	Description
0	Disable the timer restarting on TCMP1
1	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)
4	RXEOF	Comparator 1 and timer are started at RX end of frame

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	TCMPVAL	0x00	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.

19.5.28 USARTn_TIMECMP2 - Used to Generate Interrupts and Various Delays

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								0		0x0				0x0											0x00							
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	0	RW	**Restart Timer on TCMP2**

Each TCMP2 event will reset and restart the timer

Value	Description
0	Disable the timer restarting on TCMP2
1	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)
4	RXEOF	Comparator 2 and timer are started at RX end of frame

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	TCMPVAL	0x00	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.

19.5.29 USARTn_ROUTEPEN - I/O Routing Pin 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																											0	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											RTSPEN	CTSPEN	CLKPEN	CSPEN	TXPEN	RXPEN

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	RTSPEN	0	RW	**RTS Pin Enable**

When set, the RTS pin of the USART is enabled.

Value	Description
0	The USn_RTS pin is disabled
1	The USn_RTS pin is enabled

Bit	Name	Reset	Access	Description
4	CTSPEN	0	RW	**CTS Pin Enable**

When set, the CTS pin of the USART is enabled.

Value	Description
0	The USn_CTS pin is disabled
1	The USn_CTS pin is enabled

Bit	Name	Reset	Access	Description
3	CLKPEN	0	RW	**CLK Pin Enable**

When set, the CLK pin of the USART is enabled.

Value	Description
0	The USn_CLK pin is disabled
1	The USn_CLK pin is enabled

Bit	Name	Reset	Access	Description
2	CSPEN	0	RW	**CS Pin Enable**

When set, the CS pin of the USART is enabled.

Value	Description
0	The USn_CS pin is disabled
1	The USn_CS pin is enabled

Bit	Name	Reset	Access	Description
1	TXPEN	0	RW	**TX Pin Enable**

When set, the TX/MOSI pin of the USART is enabled

Value	Description
0	The U(S)n_TX (MOSI) pin is disabled
1	The U(S)n_TX (MOSI) pin is enabled

Bit	Name	Reset	Access	Description
0	RXPEN	0	RW	**RX Pin Enable**

When set, the RX/MISO pin of the USART is enabled.

Value	Description
0	The U(S)n_RX (MISO) pin is disabled
1	The U(S)n_RX (MISO) pin is enabled

19.5.30 USARTn_ROUTELOC0 - I/O Routing Location 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			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			CLKLOC								CSLOC								TXLOC								RXLOC					

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	CLKLOC	0x00	RW	**I/O Location**

Decides the location of the USART CLK pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6

23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:16	CSLOC	0x00	RW	**I/O Location**

Decides the location of the USART CS pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6

15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	TXLOC	0x00	RW	**I/O Location**

Decides the location of the USART TX pin.

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2
	3	LOC3		Location 3
	4	LOC4		Location 4
	5	LOC5		Location 5
	6	LOC6		Location 6
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	RXLOC	0x00	RW	**I/O Location**
	Decides the location of the USART RX pin.			
	Value	Mode		Description
	0	LOC0		Location 0
	1	LOC1		Location 1
	2	LOC2		Location 2
	3	LOC3		Location 3
	4	LOC4		Location 4
	5	LOC5		Location 5
	6	LOC6		Location 6

19.5.31 USARTn_ROUTELOC1 - I/O Routing Location Register

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																			0x00								0x00					
Access																			RW								RW					
Name																			RTSLOC								CTSLOC					

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*		
13:8	RTSLOC	0x00	RW	**I/O Location**

Decides the location of the USART RTS pin.

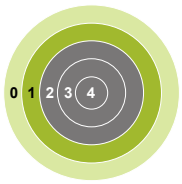
Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

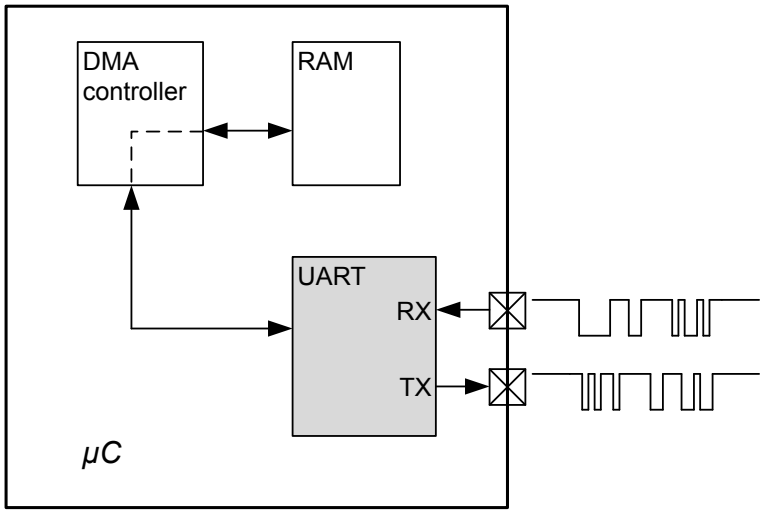
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	CTSLOC	0x00	RW	**I/O Location**

Decides the location of the USART CTS pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

20. UART - Universal Asynchronous Receiver/ Transmitter





Quick Facts

What?

The UART is capable of high-speed asynchronous serial communication

Why?

Serial communication is frequently used in embedded systems and the UART allows efficient communication with a wide range of external devices.

How?

The UART has a wide selection of operating modes, frame formats and baud rates. The multi-processor mode allows the UART 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.

20.1 Introduction

The Universal Asynchronous serial Receiver and Transmitter (UART) is a very flexible serial I/O module. It supports full- and half-duplex asynchronous UART communication.

20.2 Features

- 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 ($HFPERCLK_{UARTn}$)
- Max bit-rate
  - 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
- 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
- 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

20.3 Functional Description

The UART is functionally equivalent to the USART with the exceptions defined in the following table. The register map and register descriptions are equal to those of the USART. See the USART chapter for detailed information on the operation of the UART.

Table 20.1. UART Limitations

Feature	Limitation
Synchronous operation	Not available. SYNC, CSMA, SMSDELAY, SSSEARLY, CSINV, CPOL and CPHA in USARTn_CTRL, and MASTEREN in USARTn_STATUS are always 0.
Transmission direction	Always LSB first. MSBF in USARTn_CTRL is always 0.
Chip-select	Not available. AUTOCS in USARTn_CTRL is always 0.
SmartCard mode	Not available. SCMODE in USARTn_CTRL is always 0.
IrDA	Not available. IREN in USARTn_IRCTRL is always 0.

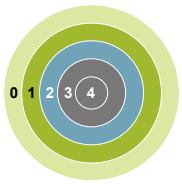
20.4 Register Map

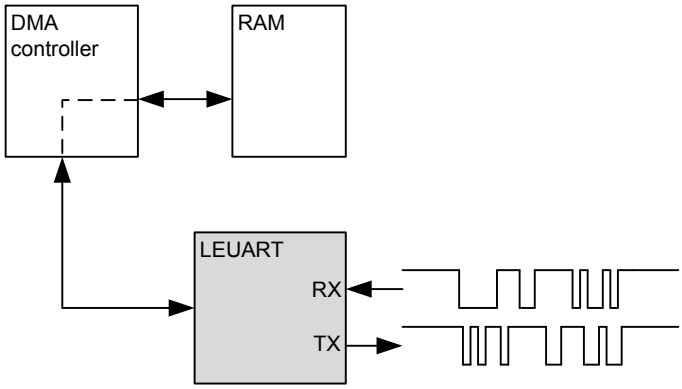
The register map of the UART is equivalent to the register map of the USART. See the USART chapter for complete information.

20.5 Register Description

The register description of the UART is equivalent to the register description of the USART except the limitations mentioned in Table 20.1 UART Limitations on page 789. See the USART chapter for complete information.

21. LEUART - Low Energy Universal Asynchronous Receiver/Transmitter





Quick Facts

What?

The LEUART provides full UART communication using a low frequency 32.768 kHz clock, and has special features for communication without CPU intervention.

Why?

It allows UART communication to be performed in low energy modes, using only a few µA during active communication and only 150 nA when waiting for incoming data.

How?

A low frequency clock signal allows communication with less energy. Using DMA, the LEUART can transmit and receive data with minimal CPU intervention. Special UART-frames can be configured to help control the data flow, further automating data transmission.

21.1 Introduction

The unique Low Energy UART (LEUART) is a UART that allows two-way UART communication on a strict power budget. Only a 32.768 kHz clock is needed to allow UART communication up to 9600 baud.

Even when the system is in low energy mode EM2 DeepSleep (with most core functionality turned off), the LEUART can wait for an incoming UART frame while having an extremely low energy consumption. When a UART frame is completely received, 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.

Received data can optionally be blocked until a configurable start frame is detected. A signal frame can be configured to generate an interrupt indicating the end of a data transmission. The start frame and signal frame can be used in combination to handle higher level communication protocols.

Similarly, data can be transmitted in EM2 DeepSleep either on a frame-by-frame basis with data from the CPU or through use of the DMA.

The LEUART includes all necessary hardware support to make asynchronous serial communication possible with minimal software overhead and low energy consumption.

21.2 Features

- Low energy asynchronous serial communications
- Full/half duplex communication
- Separate TX / RX enable
- Separate double buffered transmit buffer and receive buffer
- Programmable baud rate, generated as a fractional division of the LFBCLK
  - Supports baud rates from 300 baud to 9600 baud
- Can use a high frequency clock source for even higher baud rates
- Configurable number of data bits: 8 or 9 (plus parity bit, if enabled)
- Configurable parity: off, even or odd
  - HW parity bit generation and check
- Configurable number of stop bits, 1 or 2
- Capable of sleep-mode wake-up on received frame
  - Either wake-up on any received byte or
  - Wake up only on specified start and signal frames
- Supports transmission and reception in EM0 Active, EM1 Sleep and EM2 DeepSleep with
  - Full DMA support
  - Specified start-frame can start reception automatically
- IrDA modulator (pulse generator, pulse extender)
- Multi-processor mode
- Loopback mode
  - Half duplex communication
  - Communication debugging
- PRS RX input

21.3 Functional Description

An overview of the LEUART module is shown in Figure 21.1 LEUART Overview on page 792.

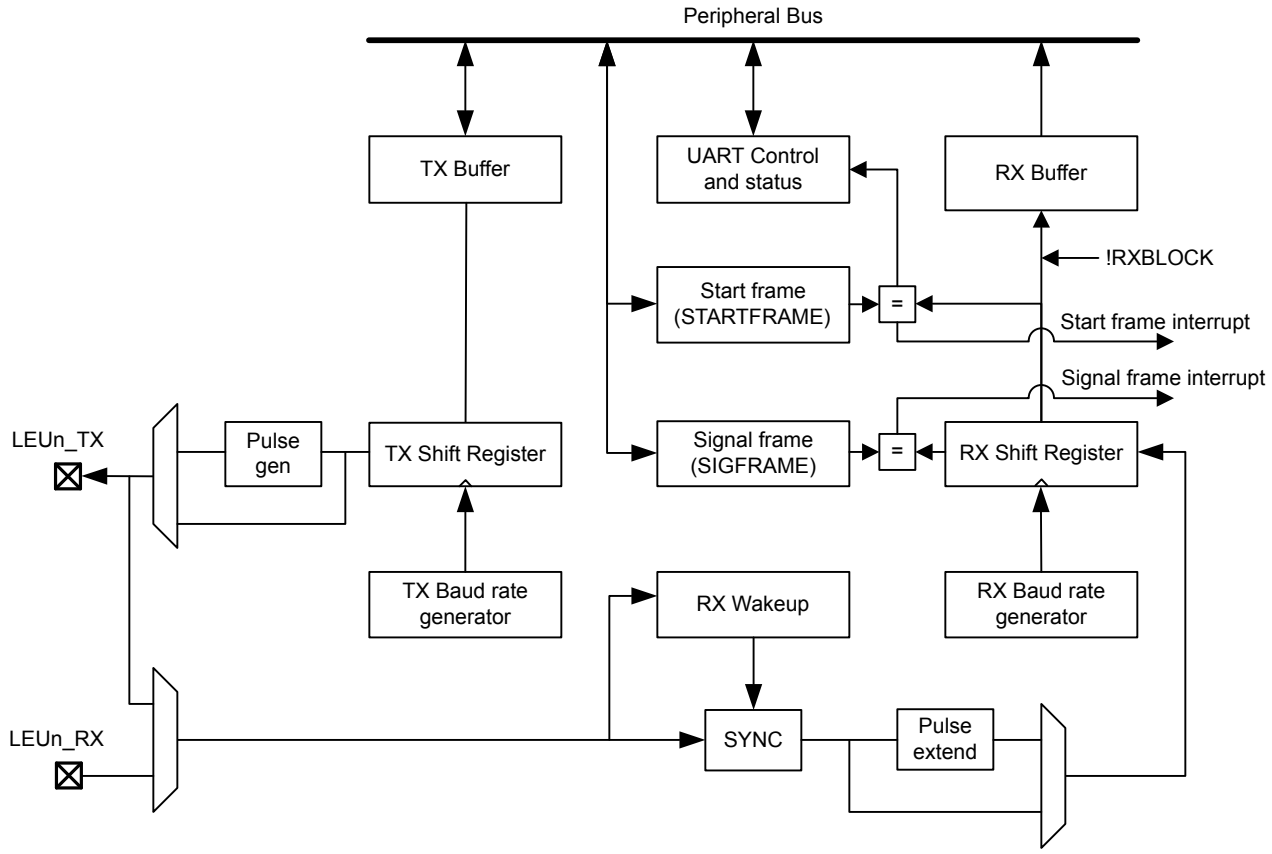


Figure 21.1. LEUART Overview

21.3.1 Frame Format

The frame format used by the LEUART 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 8 or 9 data bits and an optional parity bit. The data is transmitted with the least significant bit first. Finally, a number of stop-bits, where the line is driven high, end the frame. The frame format is shown in Figure 21.2 LEUART Asynchronous Frame Format on page 793.

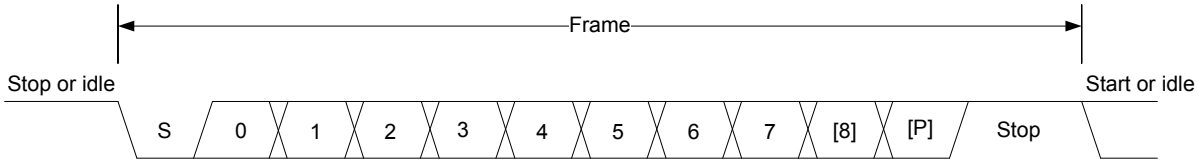


Figure 21.2. LEUART Asynchronous Frame Format

The number of data bits in a frame is set by DATABITS in LEUARTn_CTRL, and the number of stop-bits is set by STOPBITS in LEUARTn_CTRL. Whether or not a parity bit should be included, and whether it should be even or odd is defined by PARITY in LEUARTn_CTRL. For communication to be possible, all parties of an asynchronous transfer must agree on the frame format being used.

The frame format used by the LEUART can be inverted by setting INV in LEUARTn_CTRL. This affects the entire frame, resulting in a low idle state, a high start-bit, inverted data and parity bits, and low stop-bits. INV should only be changed while the receiver is disabled.

21.3.1.1 Parity Bit Calculation and Handling

Hardware automatically inserts parity bits into outgoing frames and checks the parity bits of incoming frames. The possible parity modes are defined in Table 21.1 LEUART Parity Bit on page 793. 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. When parity bits are disabled, which is the default configuration, the parity bit is omitted.

Table 21.1. LEUART Parity Bit

PARITY [1:0]	Description
00	No parity (default)
01	Reserved
10	Even parity
11	Odd parity

See 21.3.5.4 Parity Error for more information on parity bit handling.

21.3.2 Clock Source

The LEUART clock source is selected by the LFB bit field the CMU_LFBCLKSEL register. The clock is prescaled by the LEUARTn bitfield in the CMU_LFBPRESC0 register and enabled by the LEUARTn bit in the CMU_LFBCLKEN0. See Figure 10.2 CMU Overview - Low Frequency Portion on page 363 for a diagram of the clocking structure.

To use this module, the LE interface clock must be enabled in CMU_HFBUSCLKEN0, in addition to the module clock.

21.3.3 Clock Generation

The LEUART clock defines the transmission and reception data rate. The clock generator employs a fractional clock divider to allow baud rates that are not attainable by integral division of the 32.768 kHz clock that drives the LEUART.

The clock divider used in the LEUART is a 14-bit value, with a 9-bit integral part and a 5-bit fractional part. The baud rate of the LEUART is given by :

$$br = fLEUARTn / (1 + LEUARTn\_CLKDIV / 256)$$

Figure 21.3. LEUART Baud Rate Equation

where fLEUARTn is the clock frequency supplied to the LEUART. The value of LEUARTn_CLKDIV thus defines the baud rate of the LEUART. The integral part of the divider is right-aligned in the upper 24 bits of LEUARTn_CLKDIV and the fractional part is left-aligned in the lower 8 bits. The divider is thus a 256th of LEUARTn_CLKDIV as seen in the equation.

As an example let us assume fLEUART = 22.5 kHz and the value of DIV in LEUARTn_CLKDIV is 0x0028 (LEUARTn_CLKDIV = 0x00000140). The baud rate = 22.5 kHz / (1 + 0x140 / 256) = 22.5 kHz / 2.25 = 10 kHz.

For a desired baud rate $br_{DESIRED}$, LEUARTn_CLKDIV can be calculated by using:

$$LEUARTn\_CLKDIV = 256 \times (fLEUARTn/br_{DESIRED} - 1)$$

Figure 21.4. LEUART CLKDIV Equation

It's important to note that this equation results in a 32bit value for the LEUARTn_CLKDIV register but only bits [16:3] are valid and all others must be 0. For example if we have a 32 kHz clock and whish to achieve a baud rate of 10 kHz the equation above results in a LEUARTn_CLKDIV value of 0x233. However, the actual value of the register will be 0x230 since bits [2:0] cannot be set. This limits the best achievable acuracy. In this example the actual baud rate will be 32 kHz / (1+ 0x230 / 255) = 10.039 kHz instead of 32 kHz / (1+ 0x233 / 255) = 10.002 kHz.

Table 21.2 LEUART Baud Rates on page 794 lists a set of desired baud rates and the closest baud rates reachable by the LEUART with a 32.768 kHz clock source. It also shows the average baud rate error.

Table 21.2. LEUART Baud Rates

Desired baud rate	LEUARTn_CLKDIV	LEUARTn_CLKDIV/256	Actual Baud Rate	Error [%]
300	27704	108.21875	300.0217	0.01
600	13728	53.625	599.8719	-0.02
1200	6736	26.3125	1199.744	-0.02
2400	3240	12.65625	2399.487	-0.02
4800	1488	5.8125	4809.982	0.21
9600	616	2.40625	9619.963	0.21

21.3.4 Data Transmission

Data transmission is initiated by writing data to the transmit buffer using one of the methods described in 21.3.4.1 Transmit Buffer Operation. When the transmit 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 LEUARTn_CMD by setting TXEN, and disabled by setting TXDIS. 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 LEUARTn_STATUS. After a transmission, when there is no more data in the shift register or transmit buffer, the TXC flag in LEUARTn_STATUS and the TXC interrupt flag in LEUARTn_IF are set, signaling that the transmitter is idle. The TXC status flag is cleared when a new byte becomes available for transmission, but the TXC interrupt flag must be cleared by software.

21.3.4.1 Transmit Buffer Operation

A frame can be loaded into the transmit buffer by writing to LEUARTn_TXDATA or LEUARTn_TXDATAX. Using LEUARTn_TXDATA allows 8 bits to be written to the buffer. If 9 bit frames are used, the 9th bit will in that case be set to the value of BIT8DV in LEUARTn_CTRL. To set the 9th bit directly and/or use transmission control, LEUARTn_TXDATAX must be used. When writing data to the transmit buffer using LEUARTn_TXDATAX, the 9th bit written to LEUARTn_TXDATAX overrides the value in BIT8DV, and alone defines the 9th bit that is transmitted if 9-bit frames are used.

If a write is attempted to the transmit buffer when it is not empty, the TXOF interrupt flag in LEUARTn_IF is set, indicating the overflow. The data already in the buffer is in that case preserved, and no data is written.

In addition to the interrupt flag TXC in LEUARTn_IF and the status flag TXC in LEUARTn_STATUS which are set when the transmitter becomes idle, TXBL in LEUARTn_STATUS and the TXBL interrupt flag in LEUARTn_IF are used to indicate the level of the transmit buffer. Whenever the transmit buffer becomes empty, these flags are set high. Both the TXBL status flag and the TXBL interrupt flag are cleared automatically when data is written to the transmit buffer.

There is also TXIDLE status in LEUART_STATUS which can be used to detect when the transmit state machine is in the idle state.

The transmit buffer, including the TX shift register can be cleared by setting command bit CLEARTX in LEUARTn_CMD. This will prevent the LEUART 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. An overview of the operation of the transmitter is shown in Figure 21.5 LEUART Transmitter Overview on page 795.

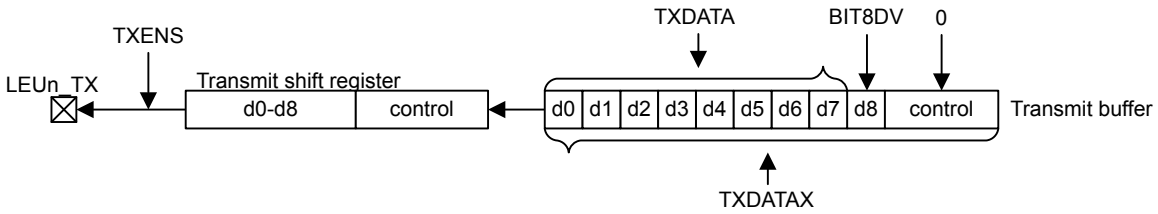


Figure 21.5. LEUART Transmitter Overview

21.3.4.2 Frame Transmission Control

The transmission control bits, which can be written using LEUARTn_TXDATAX, 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 first 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 for one bit period before the next frame is transmitted so the next start condition can be identified correctly by the recipient. Continuous breaks lasting longer than an UART frame are thus not supported by the LEUART. GPIO can be used for this. Note that when AUTOTRI in LEUARTn_CTRL is used, the transmitter is not tristated before the high-bit after the break has been transmitted.
- 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.

The transmission control bits in the LEUART cannot tristate the transmitter. This is performed automatically by hardware if AUTOTRI in LEUARTn_CTRL is set. See 21.3.7 Half Duplex Communication for more information on half duplex operation.

21.3.5 Data Reception

Data reception is enabled by setting RXEN in LEUARTn_CMD. When the receiver is enabled, it actively samples the input looking for a transition from high to low indicating the 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 buffer, making the shift register ready for another frame of data, and the receiver starts looking for another start bit. 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 receive shift register is overwritten, and the RXOF interrupt flag in LEUARTn_IF is set to indicate the buffer overflow.

The receiver can be disabled by setting the command bit RXDIS in LEUARTn_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 LEUARTn_STATUS.

The receive buffer,can be cleared by setting command bit CLEARRX in LEUARTn_CMD. This will make it avaliable for new data. Any frame currently being received will not be aborted and will become the first received frame when complete.

21.3.5.1 Receive Buffer Operation

When data becomes available in the receive buffer, the RXDATAV flag in LEUARTn_STATUS and the RXDATAV interrupt flag in LEUARTn_IF are set. Both the RXDATAV status flag and the RXDATAV interrupt flag are cleared by hardware when data is no longer available, i.e. when data has been read out of the buffer.

Data can be read from receive buffer using either LEUARTn_RXDATA or LEUARTn_RXDATAX. LEUARTn_RXDATA gives access to the 8 least significant bits of the received frame, while LEUARTn_RXDATAX must be used to get access to the 9th, most significant bit. The LEUARTn_RXDATAX register also contains status information regarding the frame.

When a frame is read from the receive buffer using LEUARTn_RXDATA or LEUARTn_RXDATAX, the frame is removed from the buffer, making room for a new one. If an attempt is done to read more frames from the buffer than what is available, the RXUF interrupt flag in LEUARTn_IF is set to signal the underflow, and the data read from the buffer is undefined.

Frames can also be read from the receive buffer without removing the data by using LEUARTn_RXDATAXP, which gives access to the frame in the buffer including control bits. Data read from this register when the receive buffer is empty is undefined. No underflow interrupt is generated by a read using LEUARTn_RXDATAXP, i.e. the RXUF interrupt flag is never set as a result of reading from LEUARTn_RXDATAXP.

An overview of the operation of the receiver is shown in Figure 21.6 LEUART Receiver Overview on page 796.

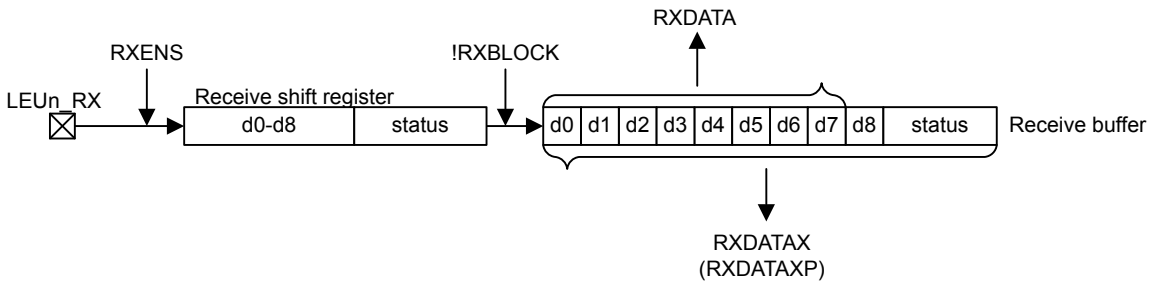


Figure 21.6. LEUART Receiver Overview

21.3.5.2 Blocking Incoming Data

When using hardware frame recognition, as detailed in 21.3.5.6 Programmable Start Frame, 21.3.5.7 Programmable Signal Frame, and 21.3.5.8 Multi-Processor Mode, 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 LEUARTn_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 bit in LEUARTn_STATUS or the RXDATAV interrupt flag in LEUARTn_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 LEUARTn_CMD and disabled by setting RXBLOCKDIS also in LEUARTn_CMD. There are two exceptions where data is loaded into the receive buffer even when RXBLOCK is set. The first is when an address frame is received when in operating in multi-processor mode as shown in 21.3.5.8 Multi-Processor Mode. The other case is when receiving a start-frame when SFUBRX in LEUARTn_CTRL is set; see 21.3.5.6 Programmable Start Frame

Frames received containing framing or parity errors will not result in the FERR and PERR interrupt flags in LEUARTn_IF being set while RXBLOCK 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 LEUARTn_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 LEUARTn_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 is set.

21.3.5.3 Data Sampling

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Except for the start-bit, only a single sample is taken of each of the incoming bits.

The length of a bit-period is given by 1 + LEUARTn_CLKDIV/256, as a number of 32.768 kHz clock periods. Let the clock cycle where a start-bit is first detected be given the index 0. The optimal sampling point for each bit in the UART frame is then given by the following equation:

$$S_{opt}(n) = n\ (1 + LEUARTn\_CLKDIV/256) + LEUARTn\_CLKDIV/512$$

Figure 21.7. LEUART Optimal Sampling Point

where n is the bit-index.

Since samples are only done on the positive edges of the 32.768 kHz clock, the actual samples are performed on the closest positive edge, i.e. the edge given by the following equation:

$$S(n) = floor(n\ x\ (1 + LEUARTn\_CLKDIV/256) + LEUARTn\_CLKDIV/512)$$

Figure 21.8. LEUART Actual Sampling Point

The sampling location will thus have jitter according to difference between S_{opt} and S. The start-bit is found at n=0, then follows the data bits, any parity bit, and the stop bits.

If the value of the start-bit is found to be high, then the start-bit is discarded, and the receiver waits for a new start-bit.

21.3.5.4 Parity Error

When the parity bit is enabled, a parity check is automatically performed on incoming frames. When a parity error is detected in a frame, the data parity error bit PERR in the frame is set, as well as the interrupt flag PERR. 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 LEUARTn_RXDATAX register.

21.3.5.5 Framing Error and Break Detection

A framing error is the result of a received 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, the framing error bit FERR in the received frame is set. The interrupt flag FERR in LEUARTn_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 LEUARTn_RXDATAX or LEUARTn_RXDATAXP registers.

21.3.5.6 Programmable Start Frame

The LEUART 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 LEUART by transmitting a given frame.

When SFUBRX in LEUARTn_CTRL is set, an incoming frame matching the frame defined in LEUARTn_STARTFRAME will result in RXBLOCK in LEUARTn_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 LEUARTn_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 buffer. 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 buffer.

An incoming start frame results in the STARTF interrupt flag in LEUARTn_IF being set, regardless of the value of SFUBRX in LEUARTn_CTRL. This allows an interrupt to be made when the start frame is detected.

When 8 data-bit frame formats are used, only the 8 least significant bits of LEUARTn_STARTFRAME are compared to incoming frames. The full length of LEUARTn_STARTFRAME is used when operating with frames consisting of 9 data bits.

Note: The receiver must be enabled for start frames to be detected. In addition, a start frame with a parity error or framing error is not detected as a start frame.

21.3.5.7 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 LEUARTn_SIGFRAME is detected by the receiver, the SIGF interrupt flag in LEUARTn_IF is set. As for start frame detection, the receiver must be enabled for signal frames to be detected.

One use of the programmable signal frame is to signal the end of a multi-frame message transmitted to the LEUART. 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 LEUART 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. The device can thus wait for data packets in EM2 DeepSleep, and only be woken up when a packet has been completely received.

A signal frame with a parity error or framing error is not detected as a signal frame.

21.3.5.8 Multi-Processor Mode

To simplify communication between multiple processors and maintain compatibility with the USART, the LEUART supports a 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 LEUARTn_CTRL is identified as an address frame. When an address frame is detected, the MPAF interrupt flag in LEUARTn_IF is set, and the address frame is loaded into the receive register. This happens regardless of the value of RXBLOCK in LEUARTn_STATUS.

Multi-processor mode is enabled by setting MPM in LEUARTn_CTRL. The mode can be used in buses with multiple slaves, allowing the slaves to be addressed using the special address frames. An addressed slave, which was previously blocking reception using RXBLOCK, would then unblock reception, receive a message from the bus master, and then block reception again, waiting for the next message. See the USART for a more detailed example.

Note: The programmable start frame functionality can be used for automatic address matching, enabling reception on a correctly configured incoming frame.

An address frame with a parity error or a framing error is not detected as an address frame. The Start, Signal, and address frames should not be set to match the same frame since each of these uses separate synchronization to the peripherial clock domain.

21.3.6 Loopback

The LEUART receiver samples LEUn_RX by default, and the transmitter drives LEUn_TX by default. This is not the only configuration however. When LOOPBK in LEUARTn_CTRL is set, the receiver is connected to the LEUn_TX pin as shown in Figure 21.9 LEUART Local Loopback on page 799. This is useful for debugging, as the LEUART can receive the data it transmits, but it is also used to allow the LEUART to read and write to the same pin, which is required for some half duplex communication modes. In this mode, the LEUn_TX pin must be enabled as an output in the GPIO.

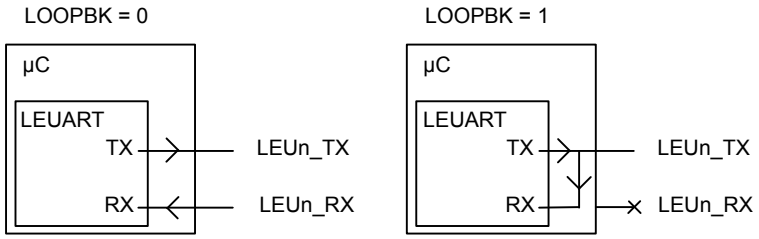


Figure 21.9. LEUART Local Loopback

21.3.7 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.

21.3.7.1 Single Data-link

In this setup, the LEUART both receives and transmits data on the same pin. This is enabled by setting LOOPBK in LEUARTn_CTRL, which connects the receiver to the transmitter output. Because they are both connected to the same line, it is important that the LEUART 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. If AUTOTRI in LEUARTn_CTRL is set, the LEUART automatically tristates LEUn_TX whenever the transmitter is inactive. It is then the responsibility of the software protocol to make sure the transmitter is not transmitting data whenever incoming data is expected.

The transmitter can also be tristated from software by configuring the GPIO pin as an input and disabling the LEUART output on LEUn_TX.

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.

21.3.7.2 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. The USART has hardware support for automatically turning the driver on and off. When using the LEUART in such a setup, the driver must be controlled by a GPIO. Figure 21.10 LEUART Half Duplex Communication with External Driver on page 800 shows an example configuration using an external driver.

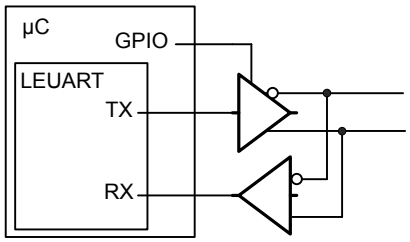


Figure 21.10. LEUART Half Duplex Communication with External Driver

21.3.7.3 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.

21.3.8 Transmission Delay

By configuring TXDELAY in LEUARTn_CTRL, the transmitter can be forced to wait a number of bit-periods from when it is ready to transmit data, to when it actually transmits the data. 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.

To route the UART TX and RX signals to a pin first select the desired pins using the RXLOC and TXLOC fields in the LEUARTn_ROUTELOC0 register. Then enable the connection using TXPEN and RXPEN in the LEUARTn_ROUTPEN register. See the device data sheet for mappings between UART locations (LOC0, LOC1, etc.) and device pins (PA0, PA1, etc.).

21.3.9 PRS RX Input

In addition to receiving data on an external pin the LEUART can be configured to receive data directly from a PRS channel by setting RX_PRS in LEUARTn_INPUT. The PRS channel used can be selected using RX_PRS_SEL in LEUARTn_INPUT. See the PRS chapter for more details on the PRS block.

For example the output of a comparator could be routed to the LEUART through the PRS to allow for receiving a signal with low peak-to-peak voltage or a significant DC offset.

21.3.10 DMA Support

The LEUART has full DMA support in energy modes EM0 Active – EM2 DeepSleep. The DMA controller can write to the transmit buffer using the registers LEUARTn_TXDATA and LEUARTn_TXDATAX, and it can read from receive buffer using the registers LEUARTn_RXDATA and LEUARTn_RXDATAX. This enables single byte transfers and 9 bit data + control/status bits transfers both to and from the LEUART. The DMA will start up the HFRCO and run from this when it is waken by the LEUART in EM2. The HFRCO is disabled once the transaction is done.

A request for the DMA controller to read from the receive buffer can come from one of the following sources:

- Receive buffer full

A write request can come from one of the following sources:

- Transmit buffer and shift register empty. No data to send.
- Transmit buffer empty

In some cases, it may be sensible to temporarily stop DMA access to the LEUART when a parity or framing error has occurred. This is enabled by setting ERRSDMA in LEUARTn_CTRL. When this bit is set, the DMA controller will not get requests from the receive buffer if a framing error or parity error is detected in the received byte. The ERRSDMA bit applies only to the RX DMA.

When operating in EM2 DeepSleep, the DMA controller must be powered up in order to perform the transfer. This is automatically performed for read operations if RXDMAWU in LEUARTn_CTRL is set and for write operations if TXDMAWU in LEUARTn_CTRL is set. To make sure the DMA controller still transfers bits to and from the LEUART in low energy modes, these bits must thus be configured accordingly.

Note: When RXDMAWU or TXDMAWU is set, the system will not be able to go to EM2 DeepSleep/EM3 Stop before all related LEUART DMA requests have been processed. This means that if RXDMAWU is set and the LEUART receives a frame, the system will not be able to go to EM2 DeepSleep/EM3 Stop before the frame has been read from the LEUART. In order for the system to go to EM2 during the last byte transmission, LEUART_CTRL_TXDMAWU must be cleared in the DMA interrupt service routine. This is because TXBL will be high during that last byte transfer.

21.3.11 Pulse Generator/ Pulse Extender

The LEUART has an optional pulse generator for the transmitter output, and a pulse extender on the receiver input. These are enabled by setting PULSEEN in LEUARTn_PULSECTRL, and with INV in LEUARTn_CTRL set, they will change the output/input format of the LEUART from NRZ to RZI as shown in Figure 21.11 LEUART - NRZ vs. RZI on page 801.

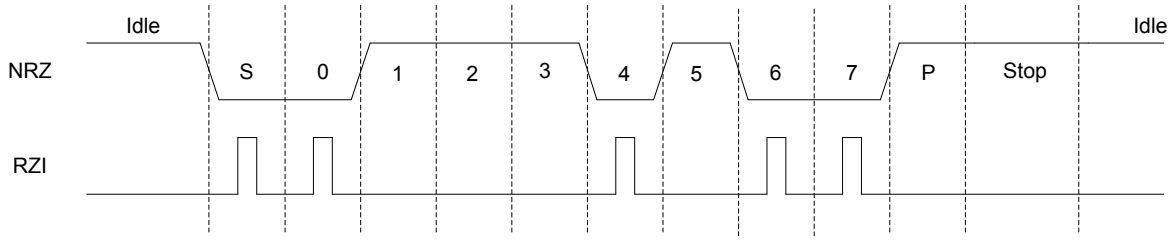


Figure 21.11. LEUART - NRZ vs. RZI

If PULSEEN in LEUARTn_PULSECTRL is set while INV in LEUARTn_CTRL is cleared, the output waveform will look like RZI shown in Figure 21.11 LEUART - NRZ vs. RZI on page 801, only inverted.

The width of the pulses from the pulse generator can be configured using PULSEW in LEUARTn_PULSECTRL. The generated pulse width is PULSEW + 1 cycles of the 32.768 kHz clock, which makes pulse width from 31.25µs to 500µs possible.

Since the incoming signal is only sampled on positive clock edges, the width of the incoming pulses must be at least two 32.768 kHz clock periods wide for reliable detection by the LEUART receiver. They must also be shorter than half a UART bit period.

At 2400 baud or lower, the pulse generator is able to generate RZI pulses compatible with the IrDA physical layer specification. The external IrDA device must generate pulses of sufficient length for successful two-way communication.

PULSEFILT in the LEUARTn_PULSECTRL register can be used to extend the minimum receive pulse width from 2 clock periods to 3 clock periods.

21.3.11.1 Interrupts

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

21.3.12 Register Access

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. Refer to 4.3 Access to Low Energy Peripherals (Asynchronous Registers) for a description on how to perform register accesses to Low Energy Peripherals.

The registers LEUARTn_FREEZE and LEUARTn_SYNCBUSY are used for synchronization of this peripheral.

21.4 Register Map

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

Offset	Name	Type	Description
0x000	LEUARTn_CTRL	RW	Control Register
0x004	LEUARTn_CMD	W1	Command Register
0x008	LEUARTn_STATUS	R	Status Register
0x00C	LEUARTn_CLKDIV	RW	Clock Control Register
0x010	LEUARTn_STARTFRAME	RW	Start Frame Register
0x014	LEUARTn_SIGFRAME	RW	Signal Frame Register
0x018	LEUARTn_RXDATAX	R(a)	Receive Buffer Data Extended Register
0x01C	LEUARTn_RXDATA	R(a)	Receive Buffer Data Register
0x020	LEUARTn_RXDATAXP	R	Receive Buffer Data Extended Peek Register
0x024	LEUARTn_TXDATAX	W	Transmit Buffer Data Extended Register
0x028	LEUARTn_TXDATA	W	Transmit Buffer Data Register
0x02C	LEUARTn_IF	R	Interrupt Flag Register
0x030	LEUARTn_IFS	W1	Interrupt Flag Set Register
0x034	LEUARTn_IFC	(R)W1	Interrupt Flag Clear Register
0x038	LEUARTn_IEN	RW	Interrupt Enable Register
0x03C	LEUARTn_PULSECTRL	RW	Pulse Control Register
0x040	LEUARTn_FREEZE	RW	Freeze Register
0x044	LEUARTn_SYNCBUSY	R	Synchronization Busy Register
0x054	LEUARTn_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x058	LEUARTn_ROUTELOC0	RW	I/O Routing Location Register
0x064	LEUARTn_INPUT	RW	LEUART Input Register

21.5 Register Description

21.5.1 LEUARTn_CTRL - Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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		0	0	0	0	0	0	0	0	0	0	0x0		0	0
Access																	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	RW
Name																	TXDELAY		TXDMAWU	RXDMAWU	BIT8DV	MPAB	MPM	SFUBRX	LOOPBK	ERRSDMA	INV	STOPBITS	PARITY		DATABITS	AUTOTRI

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*		
15:14	TXDELAY	0x0	RW	**TX Delay Transmission**

Configurable delay before new transfers. Frames sent back-to-back are not delayed.

Value	Mode	Description
0	NONE	Frames are transmitted immediately
1	SINGLE	Transmission of new frames are delayed by a single bit period
2	DOUBLE	Transmission of new frames are delayed by two bit periods
3	TRIPLE	Transmission of new frames are delayed by three bit periods

13 TXDMAWU 0 RW **TX DMA Wakeup**

Set to wake the DMA controller up when in EM2 and space is available in the transmit buffer.

Value	Description
0	While in EM2, the DMA controller will not get requests about space being available in the transmit buffer
1	DMA is available in EM2 for the request about space available in the transmit buffer

12 RXDMAWU 0 RW **RX DMA Wakeup**

Set to wake the DMA controller up when in EM2 and data is available in the receive buffer.

Value	Description
0	While in EM2, the DMA controller will not get requests about data being available in the receive buffer
1	DMA is available in EM2 for the request about data in the receive buffer

11 BIT8DV 0 RW **Bit 8 Default Value**

When 9-bit frames are transmitted, the default value of the 9th bit is given by BIT8DV. If TXDATA is used to write a frame, then the value of BIT8DV is assigned to the 9th bit of the outgoing frame. If a frame is written with TXDATAX however, the default value is overridden by the written value.

Bit	Name	Reset	Access	Description
10	MPAB	0	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.

9	MPM	0	RW	**Multi-Processor Mode**

Set to enable multi-processor mode.

Value	Description
0	The 9th bit of incoming frames have no special function
1	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

8	SFUBRX	0	RW	**Start-Frame UnBlock RX**

Clears RXBLOCK when the start-frame is found in the incoming data. The start-frame is loaded into the receive buffer.

Value	Description
0	Detected start-frames have no effect on RXBLOCK
1	When a start-frame is detected, RXBLOCK is cleared and the start-frame is loaded into the receive buffer

7	LOOPBK	0	RW	**Loopback Enable**

Set to connect receiver to LEUn_TX instead of LEUn_RX.

Value	Description
0	The receiver is connected to and receives data from LEUn_RX
1	The receiver is connected to and receives data from LEUn_TX

6	ERRSDMA	0	RW	**Clear RX DMA on Error**

When set, RX DMA requests will be cleared on framing and parity errors.

Value	Description
0	Framing and parity errors have no effect on DMA requests from the LEUART
1	RX DMA requests from the LEUART are disabled if a framing error or parity error occurs.

5	INV	0	RW	**Invert Input and Output**

Set to invert the output on LEUn_TX and input on LEUn_RX.

Value	Description
0	A high value on the input/output is 1, and a low value is 0.
1	A low value on the input/output is 1, and a high value is 0.

4	STOPBITS	0	RW	**Stop-Bit Mode**

Determines the number of stop-bits used. Only used when transmitting data. The receiver only verifies that one stop bit is present.

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	ONE		One stop-bit is transmitted with every frame
	1	TWO		Two stop-bits are transmitted with every frame
3:2	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.
1	DATABITS	0	RW	**Data-Bit Mode**
	This register sets the number of data bits.			
	Value	Mode		Description
	0	EIGHT		Each frame contains 8 data bits
	1	NINE		Each frame contains 9 data bits
0	AUTOTRI	0	RW	**Automatic Transmitter Tristate**
	When set, LEUn_TX is tristated whenever the transmitter is inactive.			
	Value			Description
	0			LEUn_TX is held high when the transmitter is inactive. INV inverts the inactive state.
	1			LEUn_TX is tristated when the transmitter is inactive

21.5.2 LEUARTn_CMD - Command Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0	0	0	0	0	0	0	0
Access																									W1	W1	W1	W1	W1	W1	W1	W1
Name																									CLEARRX	CLEARTX	RXBLOCKDIS	RXBLOCKEN	TXDIS	TXEN	RXDIS	RXEN

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	CLEARRX	0	W1	**Clear RX**
	Set to clear receive buffer and the RX shift register.			
6	CLEARTX	0	W1	**Clear TX**
	Set to clear transmit buffer and the TX shift register.			
5	RXBLOCKDIS	0	W1	**Receiver Block Disable**
	Set to clear RXBLOCK, resulting in all incoming frames being loaded into the receive buffer.			
4	RXBLOCKEN	0	W1	**Receiver Block Enable**
	Set to set RXBLOCK, resulting in all incoming frames being discarded.			
3	TXDIS	0	W1	**Transmitter Disable**
	Set to disable transmission.			
2	TXEN	0	W1	**Transmitter Enable**
	Set to enable data transmission.			
1	RXDIS	0	W1	**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	0	W1	**Receiver Enable**
	Set to activate data reception on LEUn_RX.			

21.5.3 LEUARTn_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																										1	0	1	0	0	0	0
Access																										R	R	R	R	R	R	R
Name																										TXIDLE	RXDATAV	TXBL	TXC	RXBLOCK	TXENS	RXENS

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	TXIDLE	1	R	**TX Idle**
	Set when TX is idle			
5	RXDATAV	0	R	**RX Data Valid**
	Set when data is available in the receive buffer. Cleared when the receive buffer is empty.			
4	TXBL	1	R	**TX Buffer Level**
	Indicates the level of the transmit buffer. Set when the transmit buffer is empty, and cleared when it is full.			
3	TXC	0	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.			
2	RXBLOCK	0	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.			
1	TXENS	0	R	**Transmitter Enable Status**
	Set when the transmitter is enabled.			
0	RXENS	0	R	**Receiver Enable Status**
	Set when the receiver is enabled. The receiver must be enabled for start frames, signal frames, and multi-processor address bit detection.			

21.5.4 LEUARTn_CLKDIV - Clock Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																0x0000																
Access																RW																
Name																DIV																

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*		
16:3	DIV	0x0000	RW	**Fractional Clock Divider**
	Specifies the fractional clock divider for the LEUART. Bits [7:3] are the fractional part and bits [16:8] are the integer part. The total divider is ([16:8] + [7:3]/32). To make the math easier the total divider can also be calculated as '([16:8] + [7:0]/256) where bits [0:2] will always be 0.			
2:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

21.5.5 LEUARTn_STARTFRAME - Start Frame Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																												0x000				
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	0x000	RW	**Start Frame**
	When a frame matching STARTFRAME is detected by the receiver, STARTF interrupt flag is set, and if SFUBRX is set, RXBLOCK is cleared. The start-frame is be loaded into the RX buffer.			

21.5.6 LEUARTn_SIGFRAME - Signal Frame Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																								0x000								
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	0x000	RW	**Signal Frame**
	When a frame matching SIGFRAME is detected by the receiver, SIGF interrupt flag is set.			

21.5.7 LEUARTn_RXDATAX - Receive Buffer Data Extended Register (Actionable Reads)

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																	0	0						0x000								
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	0	R	**Receive Data Framing Error**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERR	0	R	**Receive Data Parity Error**
	Set if data in buffer has a parity error.			
13:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	RXDATA	0x000	R	**RX Data**
	Use this register to access data read from the LEUART. Buffer is cleared on read access.			

21.5.8 LEUARTn_RXDATA - Receive Buffer Data Register (Actionable Reads)

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																									0x00							
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	0x00	R	**RX Data**
	Use this register to access data read from LEUART. Buffer is cleared on read access. Only the 8 LSB can be read using this register.			

21.5.9 LEUARTn_RXDATAXP - Receive Buffer Data Extended Peek 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																	0	0						0x000								
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	0	R	**Receive Data Framing Error Peek**
	Set if data in buffer has a framing error. Can be the result of a break condition.			
14	PERRP	0	R	**Receive Data Parity Error Peek**
	Set if data in buffer has a parity error.			
13:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	RXDATAP	0x000	R	**RX Data Peek**
	Use this register to access data read from the LEUART.			

21.5.10 LEUARTn_TXDATAX - Transmit Buffer Data Extended Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																	0	0	0									0x000				
Access																	W	W	W									W				
Name																	RXENAT	TXDISAT	TXBREAK									TXDATA				

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	0	W	**Enable RX After Transmission**

Set to enable reception after transmission.

Value	Description
0	The receiver is not enabled after the frame has been transmitted
1	The receiver is enabled (setting RXENS) after the frame has been transmitted

Bit	Name	Reset	Access	Description
14	TXDISAT	0	W	**Disable TX After Transmission**

Set to disable transmitter directly after transmission has competed.

Value	Description
0	The transmitter is not disabled after the frame has been transmitted
1	The transmitter is disabled (clearing TXENS) after the frame has been transmitted

Bit	Name	Reset	Access	Description
13	TXBREAK	0	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.

Value	Description
0	The specified number of stop-bits are transmitted
1	Instead of the ordinary stop-bits, 0 is transmitted to generate a break. A single stop-bit is generated after the break to allow the receiver to detect the start of the next frame

Bit	Name	Reset	Access	Description
12:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:0	TXDATA	0x000	W	**TX Data**

Use this register to write data to the LEUART. If the transmitter is enabled, a transfer will be initiated at the first opportunity.

21.5.11 LEUARTn_TXDATA - Transmit Buffer Data Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
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	0x00	W	**TX Data** This frame will be added to the transmit buffer. Only 8 LSB can be written using this register. 9th bit and control bits will be cleared.

21.5.12 LEUARTn_IF - Interrupt Flag 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																						0	0	0	0	0	0	0	0	0	1	0
Access																						R	R	R	R	R	R	R	R	R	R	R
Name																						SIGF	STARTF	MPAF	FERR	PERR	TXOF	RXUF	RXOF	RXDATAV	TXBL	TXC

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	SIGF	0	R	**Signal Frame Interrupt Flag**
	Set when a signal frame is detected. MPA, START, and SIGNAL should not be set to match the same frame since they use different synchronizers.			
9	STARTF	0	R	**Start Frame Interrupt Flag**
	Set when a start frame is detected. MPA, START, and SIGNAL should not be set to match the same frame since they use different synchronizers.			
8	MPAF	0	R	**Multi-Processor Address Frame Interrupt Flag**
	Set when a multi-processor address frame is detected. MPA, START, and SIGNAL should not be set to match the same frame since they use different synchronizers.			
7	FERR	0	R	**Framing Error Interrupt Flag**
	Set when a frame with a framing error is received while RXBLOCK is cleared.			
6	PERR	0	R	**Parity Error Interrupt Flag**
	Set when a frame with a parity error is received while RXBLOCK is cleared.			
5	TXOF	0	R	**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.			
4	RXUF	0	R	**RX Underflow Interrupt Flag**
	Set when trying to read from the receive buffer when it is empty.			
3	RXOF	0	R	**RX Overflow Interrupt Flag**
	Set when data is incoming while the receive shift register is full. The data previously in shift register is overwritten by the new data.			
2	RXDATAV	0	R	**RX Data Valid Interrupt Flag**
	Set when data becomes available in the receive buffer.			
1	TXBL	1	R	**TX Buffer Level Interrupt Flag**
	Set when space becomes available in the transmit buffer for a new frame.			
0	TXC	0	R	**TX Complete Interrupt Flag**
	Set after a transmission when both the TX buffer and shift register are empty.			

21.5.13 LEUARTn_IFS - Interrupt Flag Set 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																						0	0	0	0	0	0	0	0			0
Access																						W1	W1	W1	W1	W1	W1	W1	W1			W1
Name																						SIGF	STARTF	MPAF	FERR	PERR	TXOF	RXUF	RXOF			TXC

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	SIGF	0	W1	**Set SIGF Interrupt Flag**
	Write 1 to set the SIGF interrupt flag			
9	STARTF	0	W1	**Set STARTF Interrupt Flag**
	Write 1 to set the STARTF interrupt flag			
8	MPAF	0	W1	**Set MPAF Interrupt Flag**
	Write 1 to set the MPAF interrupt flag			
7	FERR	0	W1	**Set FERR Interrupt Flag**
	Write 1 to set the FERR interrupt flag			
6	PERR	0	W1	**Set PERR Interrupt Flag**
	Write 1 to set the PERR interrupt flag			
5	TXOF	0	W1	**Set TXOF Interrupt Flag**
	Write 1 to set the TXOF interrupt flag			
4	RXUF	0	W1	**Set RXUF Interrupt Flag**
	Write 1 to set the RXUF interrupt flag			
3	RXOF	0	W1	**Set RXOF Interrupt Flag**
	Write 1 to set the RXOF interrupt flag			
2:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	TXC	0	W1	**Set TXC Interrupt Flag**
	Write 1 to set the TXC interrupt flag			

21.5.14 LEUARTn_IFC - Interrupt Flag Clear 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																						0	0	0	0	0	0	0	0			0
Access																						(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1			(R)W1
Name																						SIGF	STARTF	MPAF	FERR	PERR	TXOF	RXUF	RXOF			TXC

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	SIGF	0	(R)W1	**Clear SIGF Interrupt Flag**
	Write 1 to clear the SIGF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	STARTF	0	(R)W1	**Clear STARTF Interrupt Flag**
	Write 1 to clear the STARTF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	MPAF	0	(R)W1	**Clear MPAF Interrupt Flag**
	Write 1 to clear the MPAF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	FERR	0	(R)W1	**Clear FERR Interrupt Flag**
	Write 1 to clear the FERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
6	PERR	0	(R)W1	**Clear PERR Interrupt Flag**
	Write 1 to clear the PERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
5	TXOF	0	(R)W1	**Clear TXOF Interrupt Flag**
	Write 1 to clear the TXOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
4	RXUF	0	(R)W1	**Clear RXUF Interrupt Flag**
	Write 1 to clear the RXUF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	RXOF	0	(R)W1	**Clear RXOF Interrupt Flag**
	Write 1 to clear the RXOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	TXC	0	(R)W1	**Clear TXC Interrupt Flag**
	Write 1 to clear the TXC interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

21.5.15 LEUARTn_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																						0	0	0	0	0	0	0	0	0	0	0
Access																						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																						SIGF	STARTF	MPAF	FERR	PERR	TXOF	RXUF	RXOF	RXDATAV	TXBL	TXC

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	SIGF	0	RW	**SIGF Interrupt Enable**
	Enable/disable the SIGF interrupt			
9	STARTF	0	RW	**STARTF Interrupt Enable**
	Enable/disable the STARTF interrupt			
8	MPAF	0	RW	**MPAF Interrupt Enable**
	Enable/disable the MPAF interrupt			
7	FERR	0	RW	**FERR Interrupt Enable**
	Enable/disable the FERR interrupt			
6	PERR	0	RW	**PERR Interrupt Enable**
	Enable/disable the PERR interrupt			
5	TXOF	0	RW	**TXOF Interrupt Enable**
	Enable/disable the TXOF interrupt			
4	RXUF	0	RW	**RXUF Interrupt Enable**
	Enable/disable the RXUF interrupt			
3	RXOF	0	RW	**RXOF Interrupt Enable**
	Enable/disable the RXOF interrupt			
2	RXDATAV	0	RW	**RXDATAV Interrupt Enable**
	Enable/disable the RXDATAV interrupt			
1	TXBL	0	RW	**TXBL Interrupt Enable**
	Enable/disable the TXBL interrupt			
0	TXC	0	RW	**TXC Interrupt Enable**
	Enable/disable the TXC interrupt			

21.5.16 LEUARTn_PULSECTRL - Pulse Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																											0	0	0x0			
Access																											RW	RW	RW			
Name																											PULSEFILT	PULSEEN	PULSEW			

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	PULSEFILT	0	RW	**Pulse Filter**
	Enable a one-cycle pulse filter for pulse extender			

Value	Description
0	Filter is disabled. Pulses must be at least 2 cycles long for reliable detection.
1	Filter is enabled. Pulses must be at least 3 cycles long for reliable detection.

4	PULSEEN	0	RW	**Pulse Generator/Extender Enable**
	Filter LEUART output through pulse generator and the LEUART input through the pulse extender.			
3:0	PULSEW	0x0	RW	**Pulse Width**
	Configure the pulse width of the pulse generator as a number of 32.768 kHz clock cycles.			

21.5.17 LEUARTn_FREEZE - Freeze 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																																0
Access																																RW
Name																																REGFREEZE

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	REGFREEZE	0	RW	**Register Update Freeze**

When set, the update of the LEUART logic from registers is postponed until this bit is cleared. Use this bit to update several registers simultaneously.

Value	Mode	Description
0	UPDATE	Each write access to a LEUART register is updated into the Low Frequency domain as soon as possible.
1	FREEZE	The LEUART is not updated with the new written value.

21.5.18 LEUARTn_SYNCBUSY - Synchronization Busy 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																									0	0	0	0	0	0	0	0
Access																									R	R	R	R	R	R	R	R
Name																									PULSECTRL	TXDATA	TXDATAX	SIGFRAME	STARTFRAME	CLKDIV	CMD	CTRL

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	PULSECTRL	0	R	**PULSECTRL Register Busy**
	Set when the value written to PULSECTRL is being synchronized.			
6	TXDATA	0	R	**TXDATA Register Busy**
	Set when the value written to TXDATA is being synchronized.			
5	TXDATAX	0	R	**TXDATAX Register Busy**
	Set when the value written to TXDATAX is being synchronized.			
4	SIGFRAME	0	R	**SIGFRAME Register Busy**
	Set when the value written to SIGFRAME is being synchronized.			
3	STARTFRAME	0	R	**STARTFRAME Register Busy**
	Set when the value written to STARTFRAME is being synchronized.			
2	CLKDIV	0	R	**CLKDIV Register Busy**
	Set when the value written to CLKDIV is being synchronized.			
1	CMD	0	R	**CMD Register Busy**
	Set when the value written to CMD is being synchronized.			
0	CTRL	0	R	**CTRL Register Busy**
	Set when the value written to CTRL is being synchronized.			

21.5.19 LEUARTn_ROUTEPEN - I/O Routing Pin 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																															0	0
Access																															RW	RW
Name																															TXPEN	RXPEN

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	0	RW	**TX Pin Enable**

When set, the TX pin of the LEUART is enabled.

Value	Description
0	The LEUn_TX pin is disabled
1	The LEUn_TX pin is enabled

Bit	Name	Reset	Access	Description
0	RXPEN	0	RW	**RX Pin Enable**

When set, the RX pin of the LEUART is enabled.

Value	Description
0	The LEUn_RX pin is disabled
1	The LEUn_RX pin is enabled

21.5.20 LEUARTn_ROUTELOC0 - I/O Routing Location 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																			0x00								0x00					
Access																			RW								RW					
Name																			TXLOC								RXLOC					

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*		
13:8	TXLOC	0x00	RW	**I/O Location**

Decides the location of the LEUART TX pin. See the device data sheet for the mapping between location and physical pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	RXLOC	0x00	RW	**I/O Location**

Decides the location of the LEUART RX pin. See the device data sheet for the mapping between location and physical pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

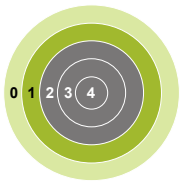
21.5.21 LEUARTn_INPUT - LEUART Input Register

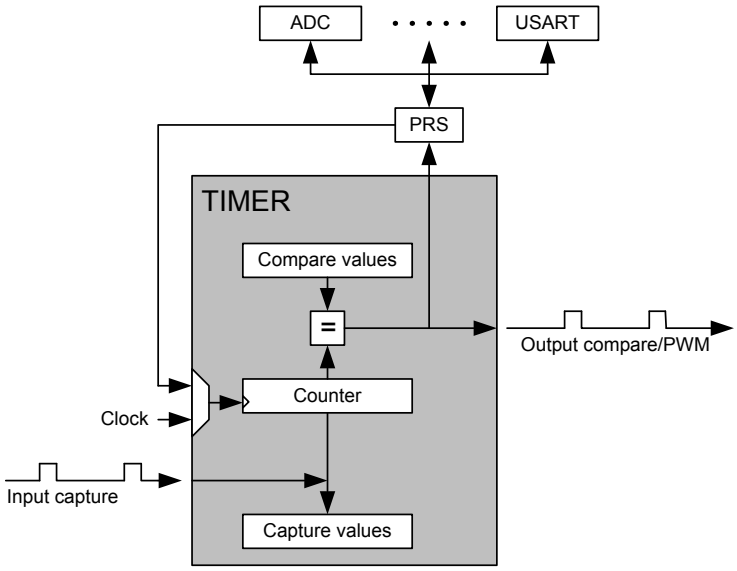
Offset	Bit Position																															
0x064	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0		0x0			
Access																											RW		RW			
Name																											RXPRS		RXPRSSEL			

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	RXPRS	0	RW	**PRS RX Enable**
	When set, the PRS channel selected as input to RX.			
4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	RXPRSSEL	0x0	RW	**RX PRS Channel Select**
	Select PRS channel as input to RX.			

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected
1	PRSCH1	PRS Channel 1 selected
2	PRSCH2	PRS Channel 2 selected
3	PRSCH3	PRS Channel 3 selected
4	PRSCH4	PRS Channel 4 selected
5	PRSCH5	PRS Channel 5 selected
6	PRSCH6	PRS Channel 6 selected
7	PRSCH7	PRS Channel 7 selected
8	PRSCH8	PRS Channel 8 selected
9	PRSCH9	PRS Channel 9 selected
10	PRSCH10	PRS Channel 10 selected
11	PRSCH11	PRS Channel 11 selected
12	PRSCH12	PRS Channel 12 selected
13	PRSCH13	PRS Channel 13 selected
14	PRSCH14	PRS Channel 14 selected
15	PRSCH15	PRS Channel 15 selected

22. TIMER/WTIMER - 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.

22.1 Introduction

The general purpose timer has 3 or 4 compare/capture channels for input capture and compare/Pulse-Width Modulation (PWM) output.

The TIMER and WTIMER peripherals are identical except for the timer width. A TIMER is 16-bits wide and a WTIMER is 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 and DTI) of each timer instance.

22.2 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
- 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
  - Configuration lock

22.3 Functional Description

An overview of the TIMER/WTIMER module is shown in Figure 22.1 TIMER/WTIMER Block Overview on page 825 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.

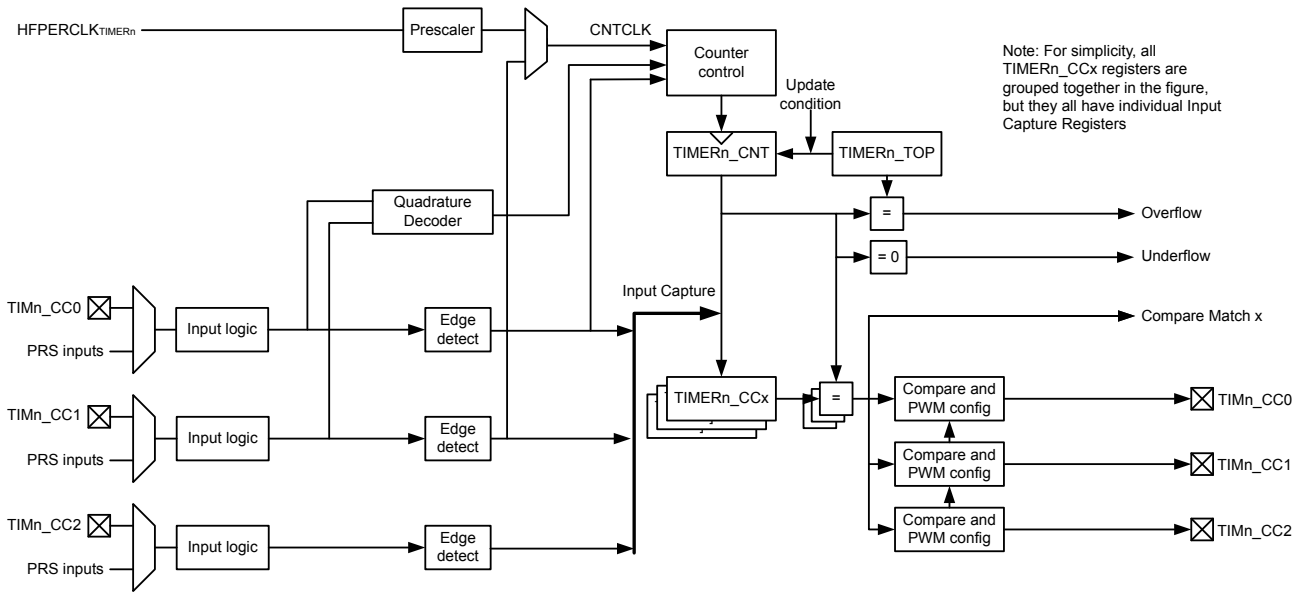


Figure 22.1. TIMER/WTIMER Block Overview

WTIMERs (Wide TIMERs) are 32-bit variants of the TIMER/WTIMER module.

22.3.1 Counter Modes

The timer consists of a counter that can be configured to the following modes:

1. Up-count: Counter counts up until it reaches the value in TIMERn_TOP, where it is reset to 0 before counting up again.
2. 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.
3. 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.
4. Quadrature Decoder: Two input channels where one determines the count direction, while the other pin triggers a clock event.

In addition, to the TIMER/WTIMER modes listed above, the TIMER/WTIMER also supports a 2x Count Mode. In this mode the counter increments/decrements by 2. The 2x Count Mode intended use is to generate 2x PWM frequency when the Compare/Capture channel is put in PWM mode. The 2x Count Mode can be enabled by setting the X2CNT bitfield in the TIMERn_CTRL register.

The counter value can be read or written by software at any time by accessing the CNT field in TIMERn_CNT.

22.3.1.1 Events

Overflow 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 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 Mode. This event is used to time updates of buffered values.

22.3.1.2 Operation

Figure 22.2 TIMER/WTIMER Hardware Timer/Counter Control on page 827 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_CTRL is set, the timer is started/stopped/reloaded (external pin or PRS) when any of the other timers are started/stopped/reloaded.

Note: TIMER0 uses a different peripheral clock compared to the other timers. Therefore it can normally not be used to start/stop/reload other timers. When using SYNC in TIMERn_CTRL for other timers make sure DYSSYNCOUT in TIMER0_CTRL is set to prevent it from triggering start/stop/reload in other timers and resulting in undesired behavior. This restriction does not apply when both HFPERCLK and HFPERBCLK are non-prescaled versions of HFCLK.

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.

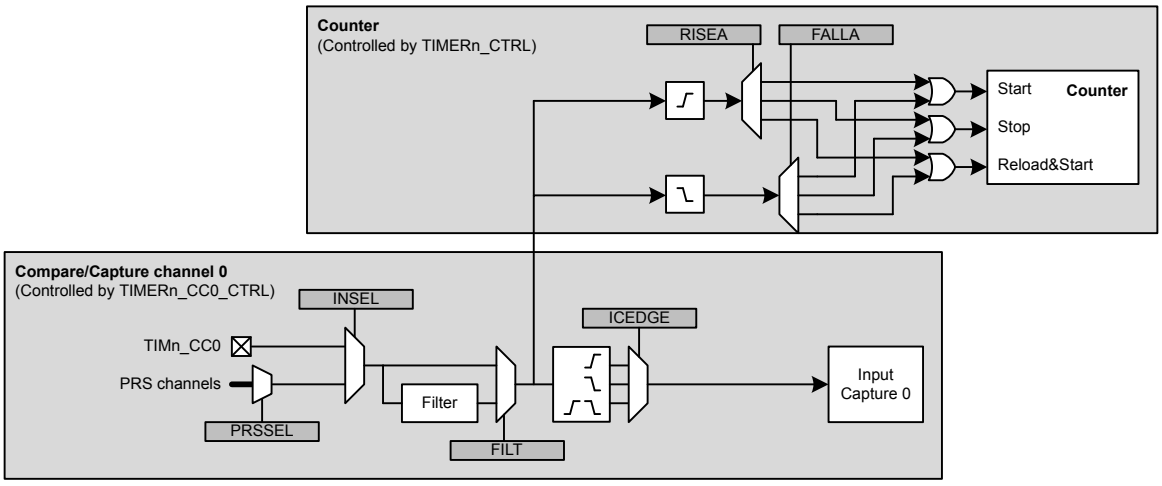


Figure 22.2. TIMER/WTIMER Hardware Timer/Counter Control

22.3.1.3 Clock Source

The counter can be clocked from several sources, which are all synchronized with the peripheral clock (HFPERCLK). See Figure 22.3 TIMER/WTIMER Clock Selection on page 828.

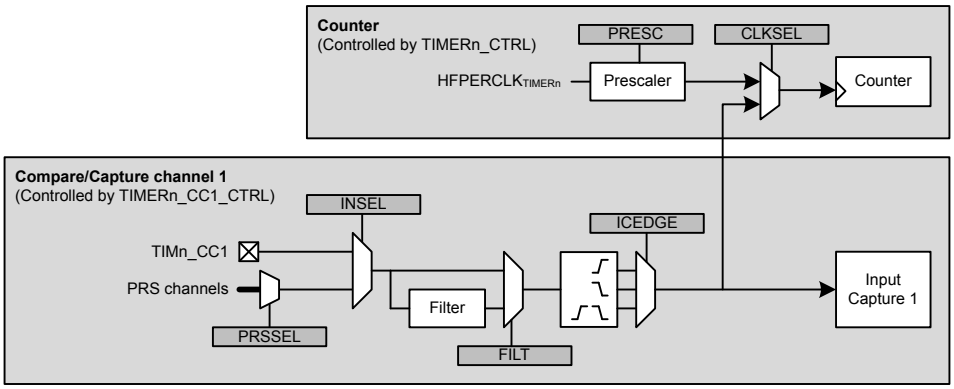


Figure 22.3. TIMER/WTIMER Clock Selection

Note: Not all TIMER instances are using the same peripheral clock. Normally the TIMER uses $HFPERCLK_{TIMERn}$, however TIMER0 supports higher frequencies and therefore uses $HFPERBCLK_{TIMER0}$. This chapter describes the general case and therefore uses $HFPERCLK_{TIMERn}$ and $f_{HFPERCLK}$, which should be interpreted as $HFPERBCLK_{TIMERn}$ and $f_{HFPERBCLK}$ for TIMER0. 10.3.1.4 HFPERCLK, HFPERBCLK, HFPERCCLK - High Frequency Peripheral Clocks shows which peripheral uses what peripheral clock.

22.3.1.4 Peripheral Clock (HFPERCLK)

The peripheral clock (HFPERCLK) can be used as a source with a configurable prescale factor of 2^PRESC, where PRESC is an integer between 0 and 10, which is set in PRESC in TIMERn_CTRL. However, if 2x Count Mode is enabled and the Compare/Capture channels are put in PWM mode, the CC output is updated on both clock edges so prescaling the peripheral clock will produce an incorrect result. The prescaler is stopped and reset when the timer is stopped.

22.3.1.5 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 $f_{HFPERCLK}/3$ when running from a pin input or a PRS input with FILT enabled in TIMERn_CCx_CTRL. When running from PRS without FILT, the frequency can be as high as $f_{HFPERCLK}$. 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.

22.3.1.6 Underflow/Overflow From Neighboring Timer

All timers are linked together (see Figure 22.4 TIMER/WTIMER Connections on page 829), allowing timers to count on overflow/underflow from the lower numbered neighbouring timers to form a 32-bit or 48-bit timer. Note that all timers must be set to same count direction and less significant timer(s) can only be set to count up or down.

Note: TIMER0 uses a different peripheral clock compared to the other timers. Therefore it can normally not be used to start/stop/reload other timers. When using SYNC in TIMERn_CTRL for other timers make sure DYSSYNCOUT in TIMER0_CTRL is set to prevent it from triggering start/stop/reload in other timers and resulting in undesired behavior. This restriction does not apply when both HFPERCLK and HFPERBCLK are non-prescaled versions of HFCLK.

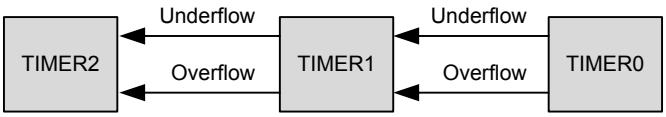


Figure 22.4. TIMER/WTIMER Connections

22.3.1.7 One-Shot Mode

By default, the counter counts continuously until it is stopped. If the OSMEN bit is set in the TIMERn_CTRL register, however, the counter is disabled by hardware on the first *update event* (see 22.3.1.1 Events). Note that when the counter is running with CC1 as clock source (0b01 in CLKSEL in TIMERn_CTRL) and OSMEN is set, a CC1 capture event will not take place on the *update event* (CC1 rising edge) that stops the timer.

22.3.1.8 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 22.5 TIMER/WTIMER TOP Value Update Functionality on page 829).

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*.

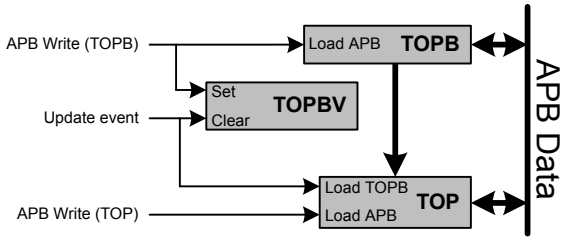


Figure 22.5. TIMER/WTIMER TOP Value Update Functionality

22.3.1.9 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 22.6 TIMER/WTIMER Quadrature Encoded Inputs on page 830).

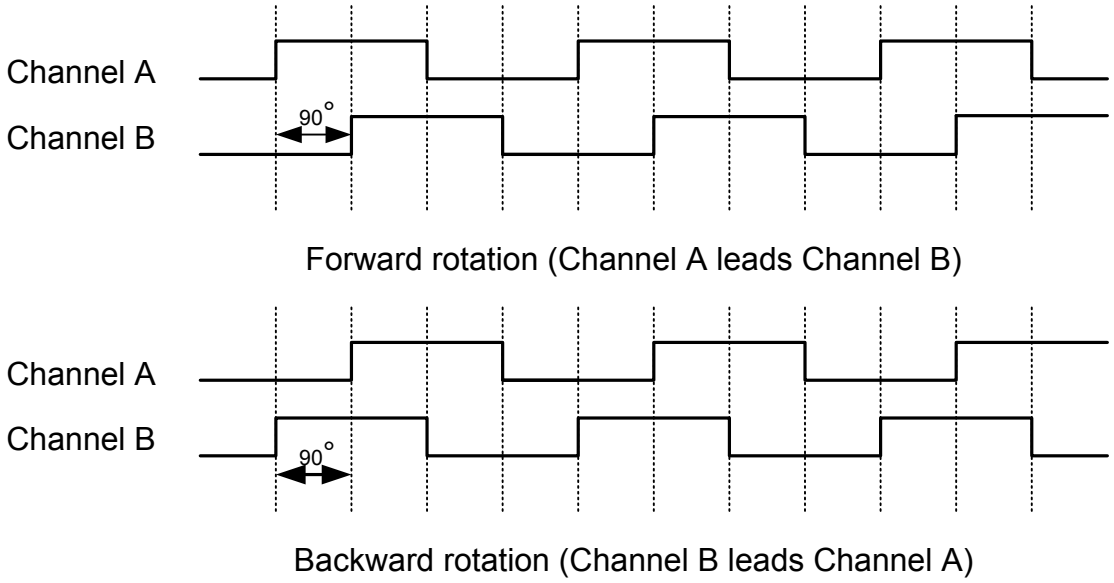


Figure 22.6. TIMER/WTIMER Quadrature Encoded Inputs

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 Quadrature Decoder Mode 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*.

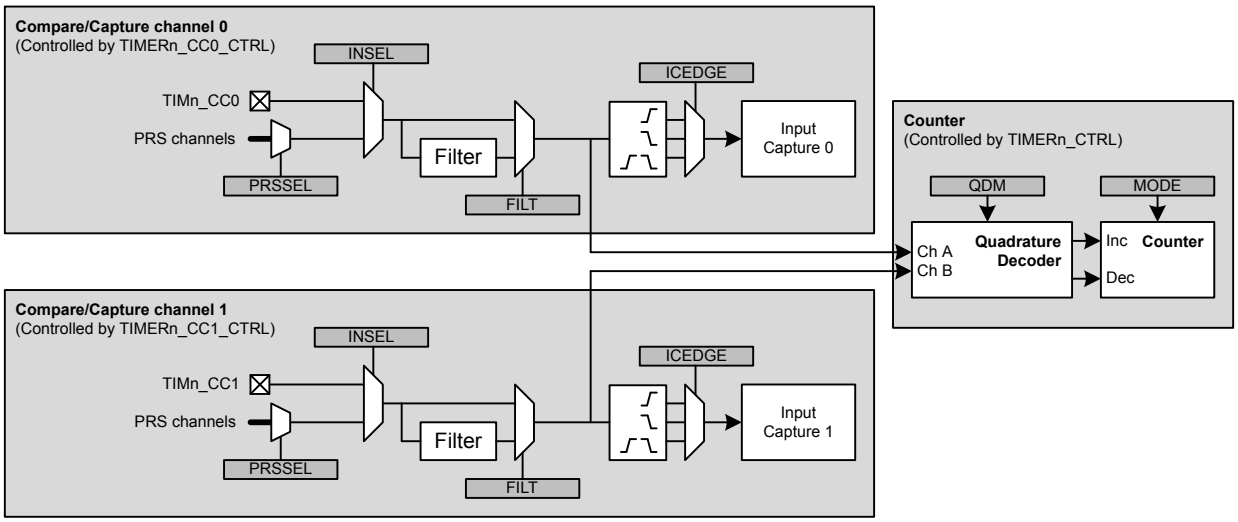


Figure 22.7. TIMER/WTIMER Quadrature Decoder Configuration

The Quadrature Decoder can be set in either X2 or X4 mode, which is configured in the QDM bit in TIMERn_CTRL. See Figure 22.7 TIMER/WTIMER Quadrature Decoder Configuration on page 830

22.3.1.10 X2 Decoding Mode

In X2 Decoding mode, the counter increments or decrements on every edge of Channel A, see Table 22.1 TIMER/WTIMER Counter Response in X2 Decoding Mode on page 831 and Figure 22.8 TIMER/WTIMER X2 Decoding Mode on page 831.

Table 22.1. TIMER/WTIMER Counter Response in X2 Decoding Mode

Channel B	Channel A	
	Rising	Falling
0	Increment	Decrement
1	Decrement	Increment

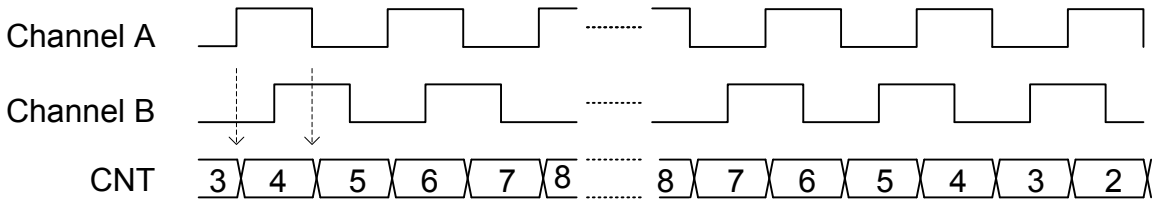


Figure 22.8. TIMER/WTIMER X2 Decoding Mode

22.3.1.11 X4 Decoding Mode

In X4 Decoding mode, the counter increments or decrements on every edge of Channel A and Channel B, see Figure 22.9 TIMER/WTIMER X4 Decoding Mode on page 831 and Table 22.2 TIMER/WTIMER Counter Response in X4 Decoding Mode on page 831.

Table 22.2. TIMER/WTIMER 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		

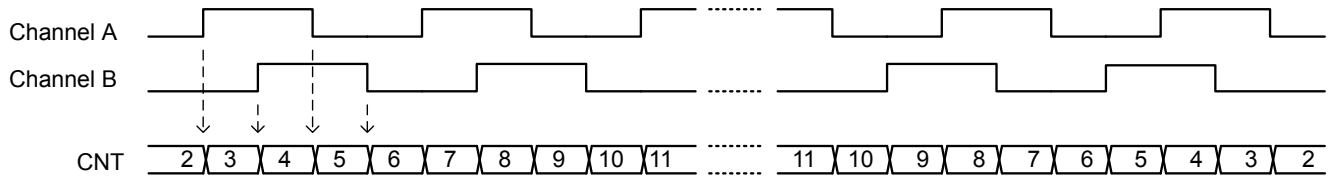


Figure 22.9. TIMER/WTIMER X4 Decoding Mode

22.3.1.12 TIMER/WTIMER Rotational Position

To calculate a position Figure 22.10 TIMER/WTIMER Rotational Position Equation on page 832 can be used.

$$pos° = (CNT/X \times N) \times 360°$$

Figure 22.10. TIMER/WTIMER Rotational Position Equation

where X = Encoding type and N = Number of pulses per revolution.

22.3.2 Compare/Capture Channels

The timer contains 3 Compare/Capture channels, which can be configured in the following modes:

1. Input Capture
2. Output Compare
3. PWM

22.3.2.1 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 22.11 TIMER/WTIMER Input Pin Logic on page 832). Compare/Capture channels 0 and 1 are the inputs for the Quadrature Decoder Mode. The input channel can be filtered before it is used, which requires the input to remain stable for 5 cycles in a row before the input is propagated to the output.

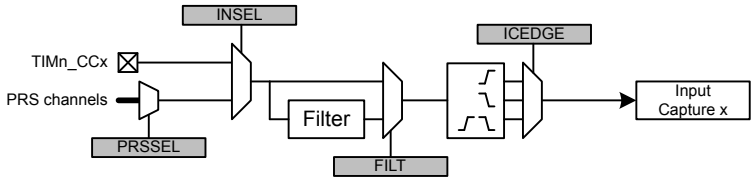


Figure 22.11. TIMER/WTIMER Input Pin Logic

22.3.2.2 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), the behavior of the Compare/Capture registers (TIMERn_CCx_CCV) and buffer registers (TIMERn_CCx_CCVB) change depending on the mode the channel is set in.

22.3.2.3 Input Capture

In Input Capture Mode, the counter value (TIMERn_CNT) can be captured in the Compare/Capture Register (TIMERn_CCx_CCV) (see Figure 22.12 TIMER/WTIMER Input Capture on page 833). The CCPOL bits in TIMERn_STATUS indicate the polarity of the edge that triggered the capture in TIMERn_CCx_CCV.

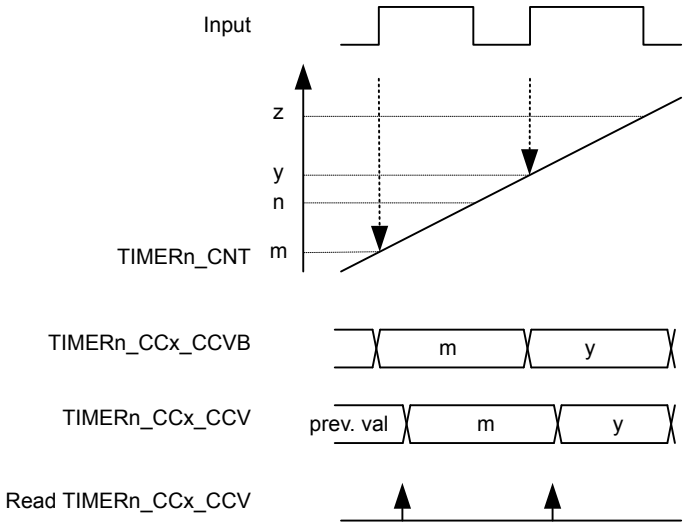


Figure 22.12. TIMER/WTIMER Input Capture

The Compare/Capture Buffer Register (TIMERn_CCx_CCVB) and the TIMERn_CCx_CCV register form double-buffered capture registers allowing two subsequent capture events to take place before a read-out is required. The first capture can always be read from TIMERn_CCx_CCV, and reading this address will load the next capture value into TIMERn_CCx_CCV from TIMERn_CCx_CCVB if it contains valid data. The CC value can be read without altering the FIFO contents by reading TIMERn_CCx_CCVP. TIMERn_CCx_CCVB can also be read without altering the FIFO contents. The ICV flag in TIMERn_STATUS indicates if there is a valid unread capture in TIMERn_CCx_CCV. In this mode, TIMERn_CCx_CCV is read-only.

In the case where a capture is triggered while both TIMERn_CCx_CCV and TIMERn_CCx_CCVB contain unread capture values, the buffer overflow interrupt flag (ICBOF in TIMERn_IF) will be set. On overflow new capture values will overwrite the value in TIMERn_CCx_CCVB and the value of TIMERn_CCx_CCV will remain unchanged. TIMERn_CCx_CCV will always contain the oldest unread value and TIMERn_CCx_CCVB will always contain the newest value.

Note: In input capture mode, the timer will only trigger interrupts when it is running.

22.3.2.4 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 select the wanted input from either external pin or PRS, see Figure 22.13 TIMER/WTIMER Period and/or Pulse width Capture on page 834. 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.

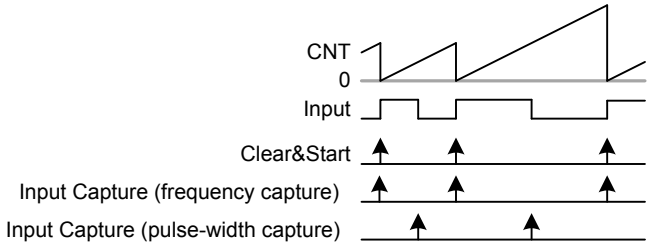


Figure 22.13. TIMER/WTIMER Period and/or Pulse width Capture

22.3.2.5 Compare

Each Compare/Capture channel contains a comparator which outputs a compare match if the contents of TIMERn_CCx_CCV matches the counter value, see Figure 22.14 TIMER/WTIMER Block Diagram Showing Comparison Functionality on page 835. 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.

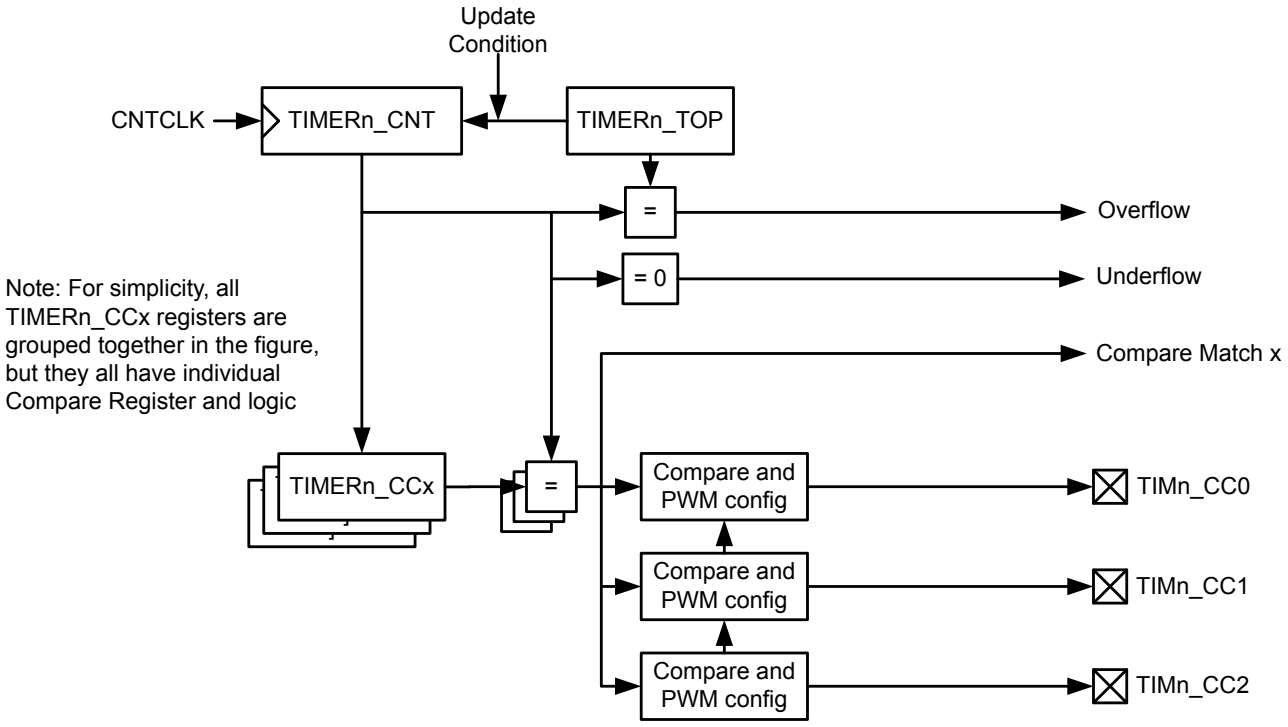


Figure 22.14. TIMER/WTIMER Block Diagram Showing Comparison Functionality

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, INSEL and FILTSEL in TIMERn_CCx_CTRL) 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_CTRL.

The COIST bit in TIMERn_CCx_CTRL 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_CTRL. 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 pulses on the output. The CC channel can be turned off by setting MODE to OFF in TIMER_CCx_CTRL. The following figure shows the output logic for the TIMER/WTIMER module.

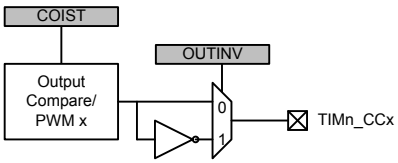


Figure 22.15. TIMER/WTIMER Output Logic

22.3.2.6 Compare Mode Registers

When running in Output Compare or PWM mode, the value in TIMERn_CCx_CCV 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_CCV can be accessed directly or through the buffer register TIMERn_CCx_CCVB, see Figure 22.16 TIMER/WTIMER Output Compare/PWM Buffer Functionality Detail on page 836. When writing to the buffer register, the value in TIMERn_CCx_CCVB will be written to TIMERn_CCx_CCV on the next *update event*. This functionality ensures glitch free PWM outputs. The CCVBV flag in TIMERn_STATUS indicates whether the TIMERn_CCx_CCVB register contains data that has not yet been written to the TIMERn_CCx_CCV register. Note that when writing 0 to TIMERn_CCx_CCVB in up-down count mode the CCV 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.

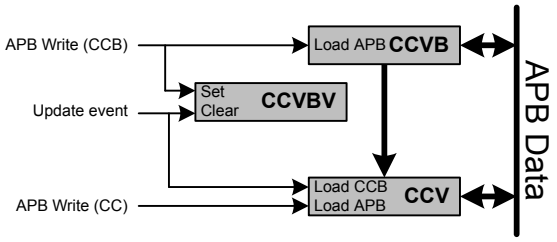


Figure 22.16. TIMER/WTIMER Output Compare/PWM Buffer Functionality Detail

22.3.2.7 Frequency Generation (FRG)

Frequency generation (see Figure 22.17 TIMER/WTIMER Up-count Frequency Generation on page 837) 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

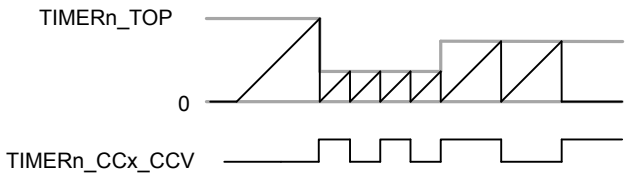


Figure 22.17. TIMER/WTIMER Up-count Frequency Generation

The output frequency is given by Figure 22.18 TIMER/WTIMER Up-count Frequency Generation Equation on page 837

$$f_{FRG} = f_{HFPERCLK} / (2^{(PRESC + 1)} \times (TOP + 1) \times 2)$$

Figure 22.18. TIMER/WTIMER Up-count Frequency Generation Equation

The figure below provides cycle accurate timing and event generation information for frequency generation.

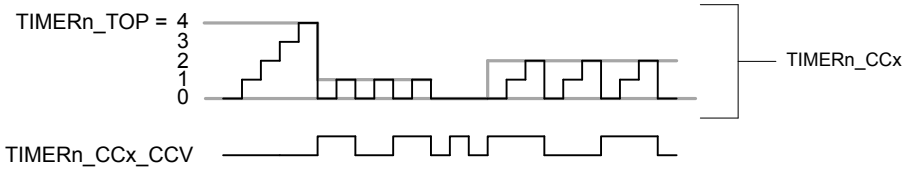


Figure 22.19. TIMER/WTIMER Up-count Frequency Generation Detail

22.3.2.8 Pulse-Width Modulation (PWM)

In PWM mode, TIMERn_CCx_CCV 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.

22.3.2.9 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 22.20 TIMER/WTIMER Up-count PWM Generation on page 838). 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_CCV. This means that a constant high output is achieved by setting TIMERn_CCx_CCV to TOP+1 or higher. The PWM resolution (in bits) is then given by Figure 22.21 TIMER/WTIMER Up-count PWM Resolution Equation on page 838.

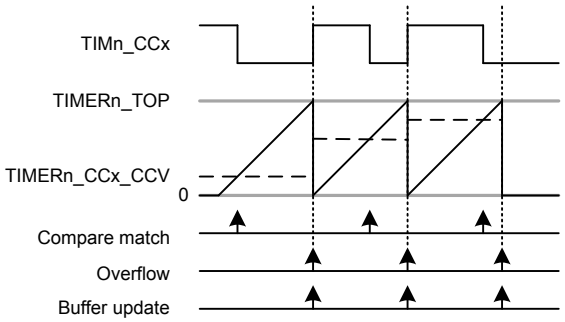


Figure 22.20. TIMER/WTIMER Up-count PWM Generation

$$R_{PWM_{up}} = \log(TOP+1)/\log(2)$$

Figure 22.21. TIMER/WTIMER Up-count PWM Resolution Equation

The PWM frequency is given by Figure 22.22 TIMER/WTIMER Up-count PWM Frequency Equation on page 838:

$$f_{PWM_{up/down}} = f_{HFPERCLK}/ (2\text{^}PRESC \times (TOP + 1))$$

Figure 22.22. TIMER/WTIMER Up-count PWM Frequency Equation

The high duty cycle is given by Figure 22.23 TIMER/WTIMER Up-count Duty Cycle Equation on page 838

$$DS_{up} = CCVx/(TOP+1)$$

Figure 22.23. TIMER/WTIMER Up-count Duty Cycle Equation

The figure below provides cycle accurate timing and event generation information for up-count mode.

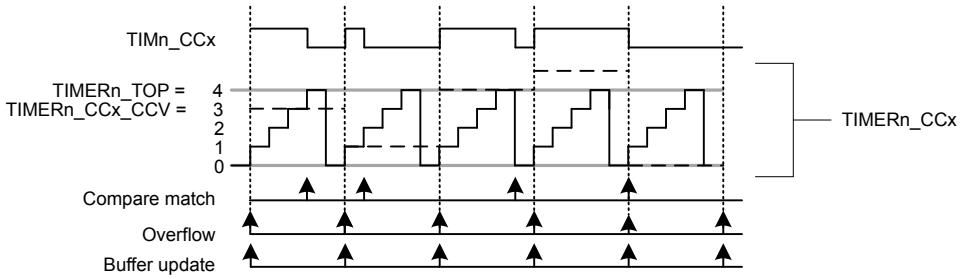


Figure 22.24. TIMER/WTIMER Up-count PWM Generation Detail

22.3.2.10 2x Count Mode (Up-count)

When the timer is set in 2x mode, the TIMER/WTIMER will count up by two. This will in effect make any odd Top value be rounded down to the closest even number. Similarly, any odd CC value will generate a match on the closest lower even value as shown in Figure 22.25 TIMER/WTIMER CC out in 2x mode on page 839

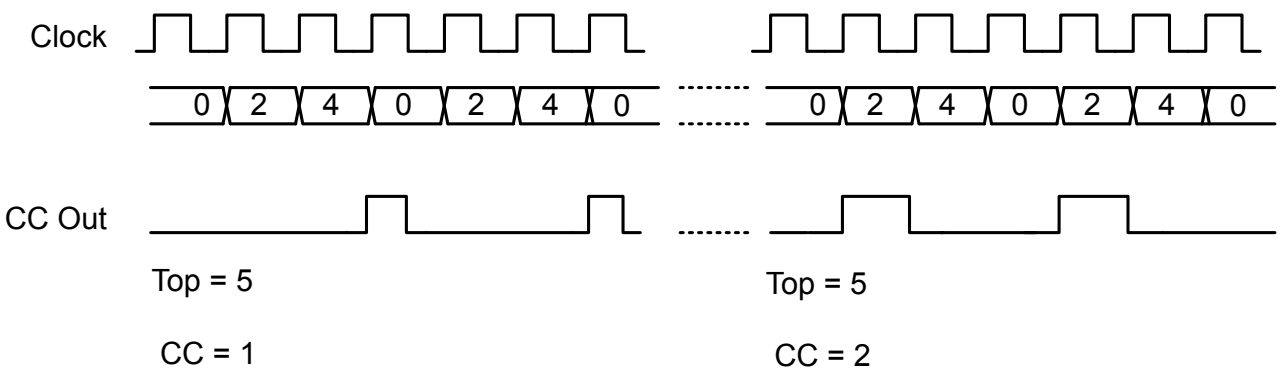


Figure 22.25. TIMER/WTIMER CC out in 2x mode

The PWM resolution is given by Figure 22.26 TIMER/WTIMER 2x PWM Resolution Equation on page 839.

$$R_{PWM_{2xmode}} = \log(TOP/2+1)/\log(2)$$

Figure 22.26. TIMER/WTIMER 2x PWM Resolution Equation

The PWM frequency is given by Figure 22.27 TIMER/WTIMER 2x Mode PWM Frequency Equation(Up-count) on page 839:

$$f_{PWM_{2xmode}} = f_{HFPERCLK}/ \text{floor}(TOP/2)+1$$

Figure 22.27. TIMER/WTIMER 2x Mode PWM Frequency Equation(Up-count)

The high duty cycle is given by Figure 22.28 TIMER/WTIMER 2x Mode Duty Cycle Equation on page 839

$$DS_{2xmode} = CCVx/((\text{floor}(TOP/2)+1)*2)$$

Figure 22.28. TIMER/WTIMER 2x Mode Duty Cycle Equation

22.3.2.11 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 22.29 TIMER/WTIMER Up/Down-count PWM Generation on page 840.The resolution (in bits) is given by Figure 22.30 TIMER/WTIMER Up/Down-count PWM Resolution Equation on page 840.

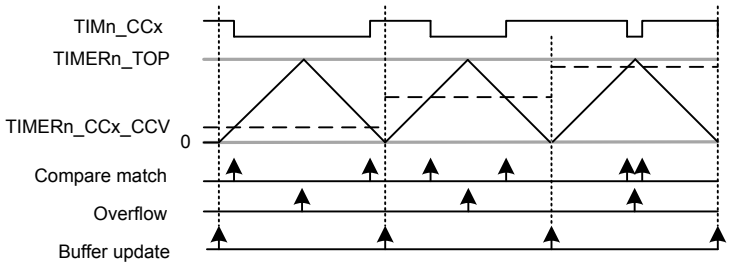


Figure 22.29. TIMER/WTIMER Up/Down-count PWM Generation

$$R_{PWM_{up/down}} = \log(TOP+1)/\log(2)$$

Figure 22.30. TIMER/WTIMER Up/Down-count PWM Resolution Equation

The PWM frequency is given by Figure 22.31 TIMER/WTIMER Up/Down-count PWM Frequency Equation on page 840:

$$f_{PWM_{up/down}} = f_{HFPERCLK}/ (2\text{^}(PRESC+1) \times TOP))$$

Figure 22.31. TIMER/WTIMER Up/Down-count PWM Frequency Equation

The high duty cycle is given by Figure 22.32 TIMER/WTIMER Up/Down-count Duty Cycle Equation on page 840

$$DS_{up/down} = CCVx/TOP$$

Figure 22.32. TIMER/WTIMER Up/Down-count Duty Cycle Equation

The figure below provides cycle accurate timing and event generation information for up-count mode.

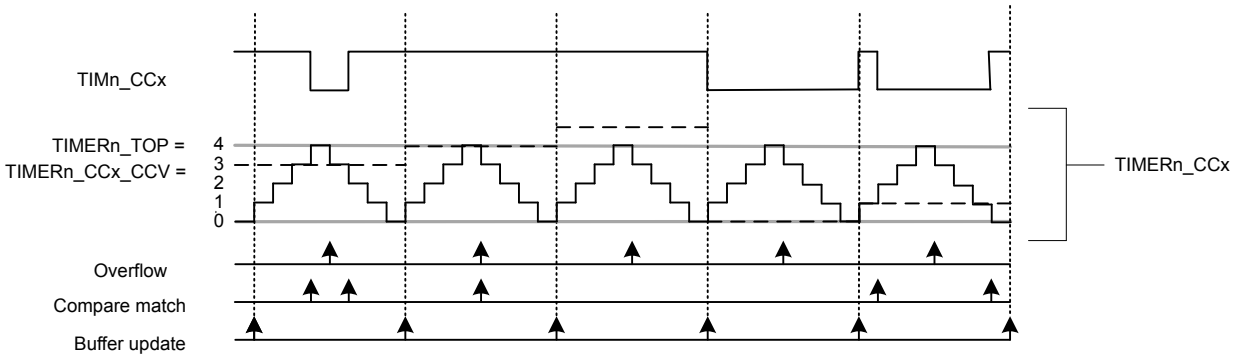


Figure 22.33. TIMER/WTIMER Up/Down-count PWM Generation

22.3.2.12 2x Count Mode (Up/Down-count)

When the timer is set in 2x mode, the TIMER/WTIMER 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 CC value will generate a match on the closest lower even value as shown in Figure 22.34 TIMER/WTIMER CC out in 2x mode on page 841

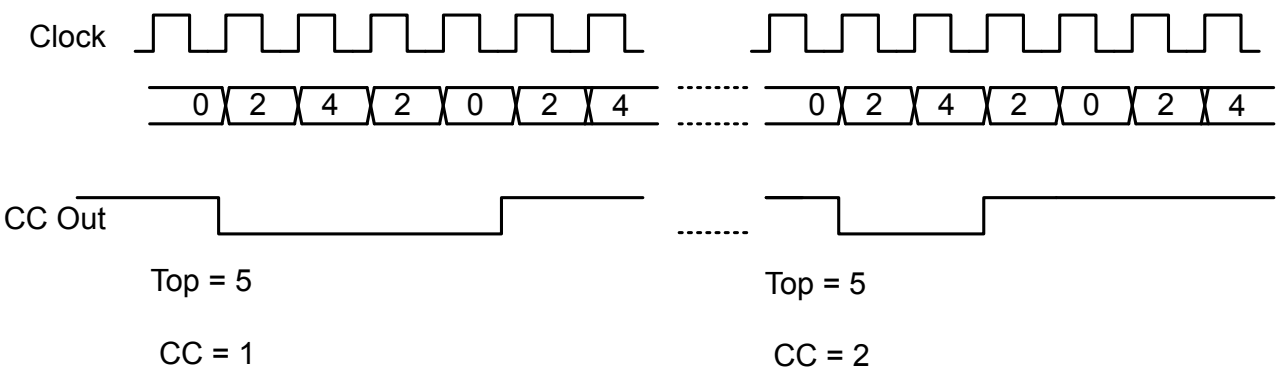


Figure 22.34. TIMER/WTIMER CC out in 2x mode

Figure 22.35 TIMER/WTIMER 2x PWM Resolution Equation on page 841.

$$R_{PWM_{2xmode}} = \log(TOP/2+1)/\log(2)$$

Figure 22.35. TIMER/WTIMER 2x PWM Resolution Equation

The PWM frequency is given by Figure 22.36 TIMER/WTIMER 2x Mode PWM Frequency Equation(Up/Down-count) on page 841:

$$f_{PWM_{2xmode}} = f_{HFPERCLK}/ (\text{floor}(TOP/2)*2)$$

Figure 22.36. TIMER/WTIMER 2x Mode PWM Frequency Equation(Up/Down-count)

The high duty cycle is given by two equations based on the CCVx values.Figure 22.37 TIMER/WTIMER 2x Mode Duty Cycle Equation for CCVx = 1 or CCVx = even on page 841 and Figure 22.38 TIMER/WTIMER 2x Mode Duty Cycle Equation for all other CCVx = odd values on page 841

$$DS_{2xmode} = (CCVx*2)/(\text{floor}(TOP/2)*4)$$

Figure 22.37. TIMER/WTIMER 2x Mode Duty Cycle Equation for CCVx = 1 or CCVx = even

$$DS_{2xmode} = (CCVx*2 - CCVx)/(\text{floor}(TOP/2)*4)$$

Figure 22.38. TIMER/WTIMER 2x Mode Duty Cycle Equation for all other CCVx = odd values

22.3.2.13 Timer Configuration Lock

To prevent software errors from making changes to the timer configuration, a configuration lock is available similar to DTI configuration Lock. Writing any value but 0xCE80 to LOCKKEY in TIMERn_LOCK results in TIMERn_CTRL, TIMERn_CMD, TIMERn_TOP, TIMERn_CNT, TIMERn_CCx_CTRL and TIMERn_CCx_CCV being locked from writing. To unlock the registers, write 0xCE80 to LOCKKEY in TIMERn_LOCK. The value of TIMERn_LOCK is 1 when the lock is active, and 0 when the registers are unlocked.

22.3.3 Dead-Time Insertion Unit

Some of the timers include a Dead-Time Insertion module suitable for motor control applications. Refer to the device data sheet to check if a timer has 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 22.39 TIMER/WTIMER Dead-Time Insertion Unit Overview on page 842.

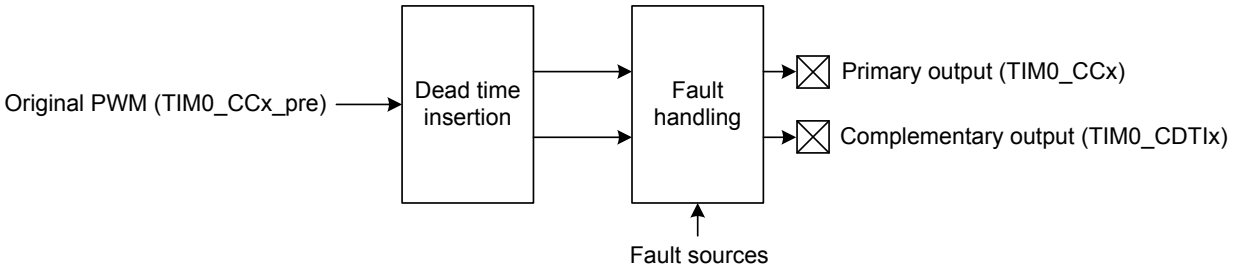


Figure 22.39. TIMER/WTIMER Dead-Time Insertion Unit Overview

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 22.40 TIMER/WTIMER Triple Half-Bridge on page 842). 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.

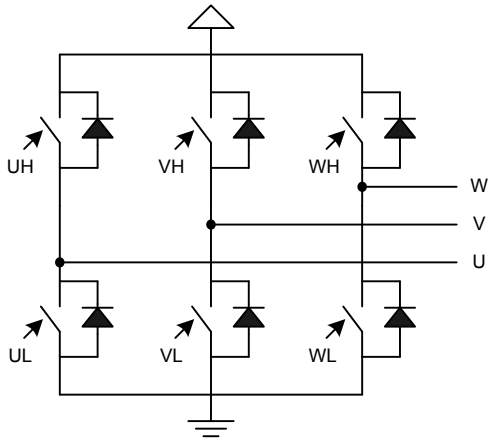


Figure 22.40. TIMER/WTIMER Triple Half-Bridge

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 22.41 TIMER/WTIMER Overview of Dead-Time Insertion Block for a Single PWM channel on page 843.

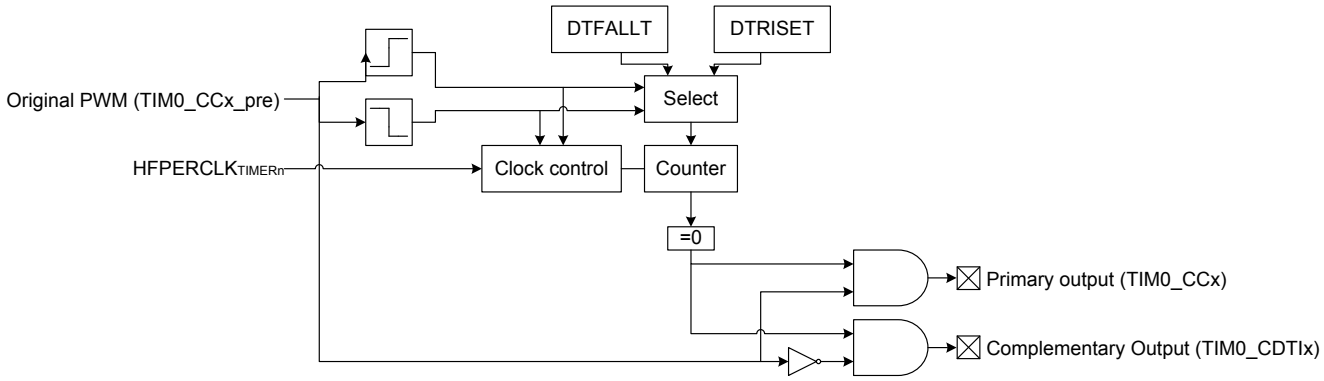


Figure 22.41. TIMER/WTIMER Overview of Dead-Time Insertion Block for a Single PWM channel

The DTI unit is enabled by setting DTEN in TIMER0_DTCTRL. 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 22.42 TIMER/WTIMER Polarity of Both Signals are Set as Active-High on page 843.

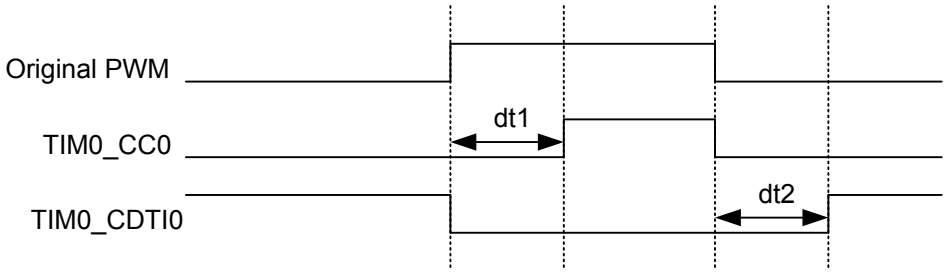


Figure 22.42. TIMER/WTIMER Polarity of Both Signals are Set as Active-High

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 $HFPERCLK_{TIMERn}$ by a configurable factor between 1 and 1024, which is set in the DTPRESC field in TIMER0_DTTIME. The rising and falling edge dead-times are configured in DTRISET and DTFALLT in TIMER0_DTTIME to any number between 1-64 $HFPERCLK_{TIMER0}$ cycles.

The DTAR and DTFATS bits in TIMER0_DTCTRL 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 to output on channel 0 to continue when the timer is stopped. DTAR effects only channel 0. See 22.3.3.2 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 programmed DTPRESC value in TIMER0_DTTIME. 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 as programmed in the DTFA field of TIMER0_DTFC register and set the DTFATS bitfield in the TIMER0_DTCTRL reg. 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 22.3. DTI Output When Timer Halted

DTAR	DTFATS	State
0	0	frozen
0	1	safe
1	0	running
1	1	running

22.3.3.1 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 22.43 TIMER/WTIMER Output Polarities on page 844.

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.

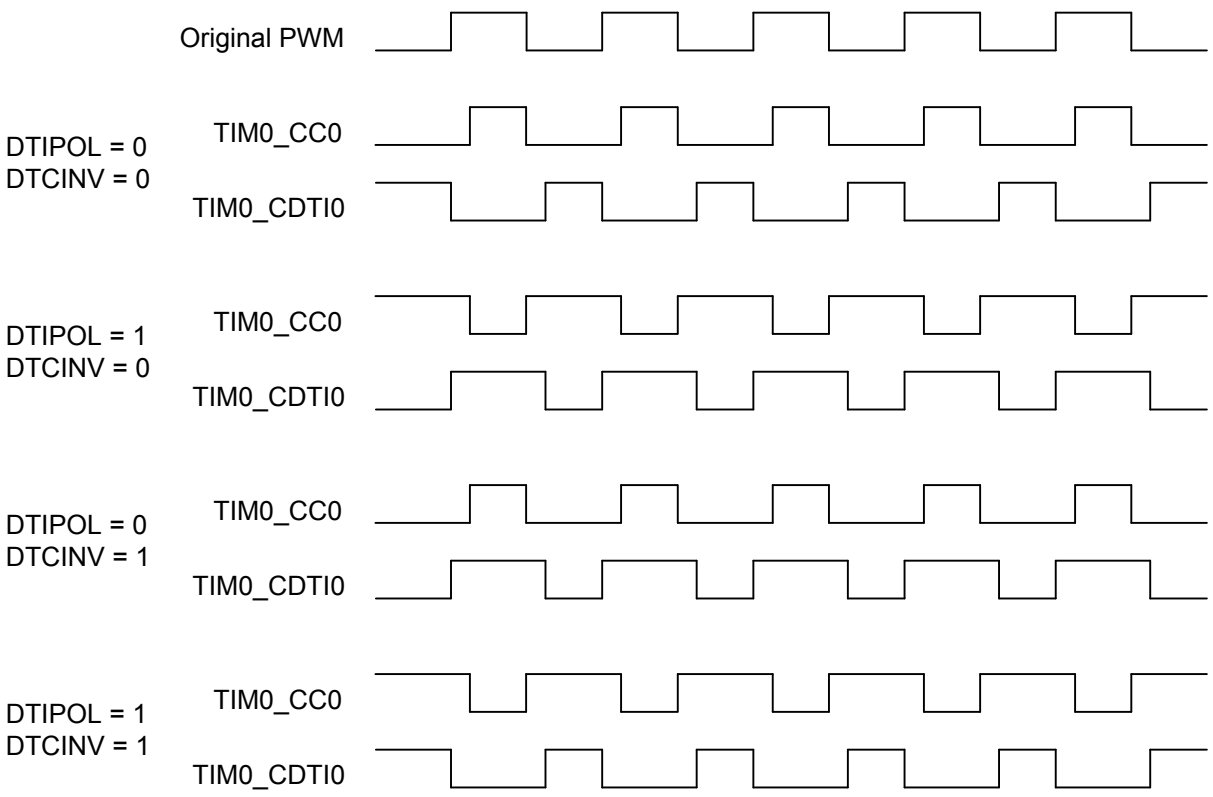


Figure 22.43. TIMER/WTIMER Output Polarities

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.

22.3.3.2 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_DTCTRL 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 by configuring DTPRSSEL in TIMER0_DTCTRL. Note that the timer must be running even when PRS is used as DTI source. However, if it is required to keep the DTI channel 0 running even when the timer is stopped, set DTAR in TIMER0_DTCTRL. When this bit is set, it uses DTPRESC=0 regardless of the value programmed in DTPRESC in TIMER0_DTTIME.

The DTI prescaler, set by DTPRESC in TIMER0_DTTIME determines the accuracy with which the DTI can insert dead-time into a PRS signal. The maximum dead-time error equals $2^{DTPRESC}$ clock cycles. With zero prescaling, the inserted dead-times are therefore accurate, but they may be inaccurate for larger prescaler settings.

22.3.3.3 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_DTFC register. Any combination of the available error sources can be selected:

- PRS source 0, determined by DTPRS0FSEL in TIMER0_DTFC
- PRS source 1, determined by DTPRS1FSEL in TIMER0_DTFC
- Debugger
- Core Lockup

One or two PRS channels can be used as an error source. When PRS source 0 is selected as an error source, DTPRS0FSEL determines which PRS channel is used for this source. DTPRS1FSEL determines which PRS channel is selected as PRS source 1. Note that for Core Lockup, the LOCKUPRDIS in RMU_CTRL must be set. Otherwise this will generate a full reset of the chip.

22.3.3.4 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 DTFACT in TIMER0_DTFC:

- 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_DTCTRL is set. This allows the DTI output to go to safe state programmed in DTFACT in TIMER0_DTFC when timer stops. When DTAR and DTFATS in TIMER0_DTCTRL 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.

22.3.3.5 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 TIMER0_DTFAULT register. If the fault source as determined by checking TIMER0_DEFAULT is the debugger alone, the outputs can be automatically restarted when the debugger exits. To enable automatic restart set DTDAS in TIMER0_DCTRL. 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.

22.3.3.6 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 results in TIMER0_DTFC, TIMER0_DTCTRL, TIMER0_DTTIME and TIMER0_ROUTE being locked from writing. To unlock the registers, write 0xCE80 to LOCKKEY in TIMER0_DTLOCK. The value of TIMER0_DTLOCK is 1 when the lock is active, and 0 when the registers are unlocked.

Note: Some of the ROUTE locations have non-interference priority. These locations prevent the use of the selected pin for other alternate functions. Thus these can be used to secure TIMER PWM outputs from software errors (i.e. another alternate function enabled to the same pin inadvertently). Therefore, it is recommended to use these locations to fully make use of the DTI Configuration Lock feature. An overview of these locations is provided in the pin map section of the device data sheet.

22.3.4 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_CTRL.

22.3.5 Interrupts, DMA and PRS Output

The timer has 3 different types of output events:

- Counter Underflow
- Counter Overflow
- Compare match or input capture (one per Compare/Capture channel)

Each of the events has its own interrupt flag. Also, there is one interrupt flag for each Compare/Capture channel which is set on buffer overflow in capture mode. Buffer overflow happens when a new capture pushes an old unread capture out of the TIMERn_CCx_CCV/TIMERn_CCx_CCVB register pair.

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 will also lead to a one $HFPERCLK_{TIMERn}$ cycle high pulse on individual PRS outputs. Setting PRSOCNF to LEVEL in TIMERn_CCx_CTRL will make the compare match PRS output follow the compare match output, instead of outputting one $HFPERCLK_{TIMERn}$ cycle high pulse. 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 22.4 TIMER/WTIMER DMA Events on page 846. 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_CTRL, 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 that does not target the CCV or CCVB register.

Table 22.4. TIMER/WTIMER DMA Events

Event	Acknowledge/Clear
Underflow/Overflow	Read or write to TIMERn_CNT or TIMERn_TOPB
CC 0	Read or write to TIMERn_CC0_CCV or TIMERn_CC0_CCVB
CC 1	Read or write to TIMERn_CC1_CCV or TIMERn_CC1_CCVB
CC 2	Read or write to TIMERn_CC2_CCV or TIMERn_CC2_CCVB

22.3.6 GPIO Input/Output

The TIMn_CCx inputs/outputs and TIM0_CDTIx outputs are accessible as alternate functions through GPIO. Each pin connection can be enabled/disabled separately by setting the corresponding CCxPEN or CDTIxPEN bits in TIMERn_ROUTE. The LOCATION bits in the same register can be used to move all enabled pins to alternate pins. See the device data sheet for the mapping between block locations (LOC0, LOC1, etc.) and actual device pins (PA0, PA1, etc.).

22.4 Register Map

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

Offset	Name	Type	Description
0x000	TIMERn_CTRL	RW	Control Register
0x004	TIMERn_CMD	W1	Command Register
0x008	TIMERn_STATUS	R	Status Register
0x00C	TIMERn_IF	R	Interrupt Flag Register
0x010	TIMERn_IFS	W1	Interrupt Flag Set Register
0x014	TIMERn_IFC	(R)W1	Interrupt Flag Clear Register
0x018	TIMERn_IEN	RW	Interrupt Enable Register
0x01C	TIMERn_TOP	RWH	Counter Top Value Register
0x020	TIMERn_TOPB	RW	Counter Top Value Buffer Register
0x024	TIMERn_CNT	RWH	Counter Value Register
0x02C	TIMERn_LOCK	RWH	TIMER Configuration Lock Register
0x030	TIMERn_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x034	TIMERn_ROUTELOC0	RW	I/O Routing Location Register
0x03C	TIMERn_ROUTELOC2	RW	I/O Routing Location Register
0x060	TIMERn_CC0_CTRL	RW	CC Channel Control Register
0x064	TIMERn_CC0_CCV	RWH(a)	CC Channel Value Register
0x068	TIMERn_CC0_CCVP	R	CC Channel Value Peek Register
0x06C	TIMERn_CC0_CCVB	RWH	CC Channel Buffer Register
...	TIMERn_CCx_CTRL	RW	CC Channel Control Register
...	TIMERn_CCx_CCV	RWH(a)	CC Channel Value Register
...	TIMERn_CCx_CCVP	R	CC Channel Value Peek Register
...	TIMERn_CCx_CCVB	RWH	CC Channel Buffer Register
0x090	TIMERn_CC3_CTRL	RW	CC Channel Control Register
0x094	TIMERn_CC3_CCV	RWH(a)	CC Channel Value Register
0x098	TIMERn_CC3_CCVP	R	CC Channel Value Peek Register
0x09C	TIMERn_CC3_CCVB	RWH	CC Channel Buffer Register
0x0A0	TIMERn_DTCTRL	RW	DTI Control Register
0x0A4	TIMERn_DTTIME	RW	DTI Time Control Register
0x0A8	TIMERn_DTFC	RW	DTI Fault Configuration Register
0x0AC	TIMERn_DTOGEN	RW	DTI Output Generation Enable Register
0x0B0	TIMERn_DTFAULT	R	DTI Fault Register
0x0B4	TIMERn_DTFAULTC	W1	DTI Fault Clear Register
0x0B8	TIMERn_DTLOCK	RWH	DTI Configuration Lock Register

22.5 Register Description

22.5.1 TIMERn_CTRL - 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			0	0	0x0										0x0			0	0		0x0		0x0		0	0	0	0	0		0x0	
Access			RW	RW	RW										RW			RW	RW		RW		RW		RW	RW	RW	RW	RW		RW	
Name			RSSCOIST	ATI	PRESC										CLKSEL			DISSYNCOUT	X2CNT		FALLA		RISEA		DMACLRACT	DEBUGRUN	QDM	OSMEN	SYNC		MODE	

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	RSSCOIST	0	RW	**Reload-Start Sets Compare Output Initial State**
	When set, compare output is set to COIST value at Reload-Start event			
28	ATI	0	RW	**Always Track Inputs**
	when set, makes CCPOL always track the polarity of the inputs			
27:24	PRESC	0x0	RW	**Prescaler Setting**
	These bits select the prescaling factor.			

Value	Mode	Description
0	DIV1	The HFPERCLK is undivided
1	DIV2	The HFPERCLK is divided by 2
2	DIV4	The HFPERCLK is divided by 4
3	DIV8	The HFPERCLK is divided by 8
4	DIV16	The HFPERCLK is divided by 16
5	DIV32	The HFPERCLK is divided by 32
6	DIV64	The HFPERCLK is divided by 64
7	DIV128	The HFPERCLK is divided by 128
8	DIV256	The HFPERCLK is divided by 256
9	DIV512	The HFPERCLK is divided by 512
10	DIV1024	The HFPERCLK is divided by 1024

23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	CLKSEL	0x0	RW	**Clock Source Select**
	These bits select the clock source for the timer.			

Value	Mode	Description
0	PRESCHFPERCLK	Prescaled HFPERCLK

Bit	Name	Reset	Access	Description
	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
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	DISSYNCOUT	0	RW	**Disable Timer From Start/Stop/Reload Other Synchronized Timers**
	When this bit is set, the Timer does not start/stop/reload other timer with SYNC bit set			
	Value			Description
	0			Timer can start/stop/reload other timers with SYNC bit set
	1			Timer cannot start/stop/reload other timers with SYNC bit set
13	X2CNT	0	RW	**2x Count Mode**
	Enable 2x count mode			
12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:10	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
9:8	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
7	DMACLRACT	0	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	0	RW	**Debug Mode Run Enable**
	Set this bit to enable timer to run in debug mode.			
	Value			Description
	0			Timer is frozen in debug mode
	1			Timer is running in debug mode

Bit	Name	Reset	Access	Description
5	QDM	0	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	0	RW	**One-shot Mode Enable**

Enable/disable one shot mode.

3	SYNC	0	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	Description
0	Timer is not started/stopped/reloaded by other timers
1	Timer is started/stopped/reloaded by other timers

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

22.5.2 TIMERn_CMD - Command 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																															0	0
Access																															W1	W1
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	0	W1	**Stop Timer**
	Set this bit to stop timer			
0	START	0	W1	**Start Timer**
	Set this bit to start timer			

22.5.3 TIMERn_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					0	0	0	0					0	0	0	0					0	0	0	0						0	0	0
Access					R	R	R	R					R	R	R	R					R	R	R	R						R	R	R
Name					CCPOL3	CCPOL2	CCPOL1	CCPOL0					ICV3	ICV2	ICV1	ICV0					CCVBV3	CCVBV2	CCVBV1	CCVBV0						TOPBV	DIR	RUNNING

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	CCPOL3	0	R	**CC3 Polarity**

In Input Capture mode, this bit indicates the polarity of the edge that triggered capture in TIMERn_CC3_CCV. In Compare/PWM mode, this bit indicates the polarity of the selected input to CC channel 3. These bits are cleared when CCMODE is written to 0b00 (Off).

Value	Mode	Description
0	LOWRISE	CC3 polarity low level/rising edge
1	HIGHFALL	CC3 polarity high level/falling edge

26	CCPOL2	0	R	**CC2 Polarity**

In Input Capture mode, this bit indicates the polarity of the edge that triggered capture in TIMERn_CC2_CCV. In Compare/PWM mode, this bit indicates the polarity of the selected input to CC channel 2. These bits are cleared when CCMODE is written to 0b00 (Off).

Value	Mode	Description
0	LOWRISE	CC2 polarity low level/rising edge
1	HIGHFALL	CC2 polarity high level/falling edge

25	CCPOL1	0	R	**CC1 Polarity**

In Input Capture mode, this bit indicates the polarity of the edge that triggered capture in TIMERn_CC1_CCV. In Compare/PWM mode, this bit indicates the polarity of the selected input to CC channel 1. These bits are cleared when CCMODE is written to 0b00 (Off).

Value	Mode	Description
0	LOWRISE	CC1 polarity low level/rising edge
1	HIGHFALL	CC1 polarity high level/falling edge

24	CCPOL0	0	R	**CC0 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

Bit	Name	Reset	Access	Description
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	ICV3	0	R	**CC3 Input Capture Valid**

This bit indicates that TIMERn_CC3_CCV contains a valid capture value. These bits are only used in input capture mode and are cleared when CCMODE is written to 0b00 (Off).

Value	Description
0	TIMERn_CC3_CCV does not contain a valid capture value(FIFO empty)
1	TIMERn_CC3_CCV contains a valid capture value(FIFO not empty)

18	ICV2	0	R	**CC2 Input Capture Valid**

This bit indicates that TIMERn_CC2_CCV contains a valid capture value. These bits are only used in input capture mode and are cleared when CCMODE is written to 0b00 (Off).

Value	Description
0	TIMERn_CC2_CCV does not contain a valid capture value(FIFO empty)
1	TIMERn_CC2_CCV contains a valid capture value(FIFO not empty)

17	ICV1	0	R	**CC1 Input Capture Valid**

This bit indicates that TIMERn_CC1_CCV contains a valid capture value. These bits are only used in input capture mode and are cleared when CCMODE is written to 0b00 (Off).

Value	Description
0	TIMERn_CC1_CCV does not contain a valid capture value(FIFO empty)
1	TIMERn_CC1_CCV contains a valid capture value(FIFO not empty)

16	ICV0	0	R	**CC0 Input Capture Valid**

This bit indicates that TIMERn_CC0_CCV contains a valid capture value. These bits are only used in input capture mode and are cleared when CCMODE is written to 0b00 (Off).

Value	Description
0	TIMERn_CC0_CCV does not contain a valid capture value(FIFO empty)
1	TIMERn_CC0_CCV contains a valid capture value(FIFO not empty)

15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	CCVBV3	0	R	**CC3 CCVB Valid**

This field indicates that the TIMERn_CC3_CCVB registers contain data which have not been written to TIMERn_CC3_CCV. These bits are only used in output compare/PWM mode and are cleared when CCMODE is written to 0b00 (Off).

Value	Description
0	TIMERn_CC3_CCVB does not contain valid data
1	TIMERn_CC3_CCVB contains valid data which will be written to TIMERn_CC3_CCV on the next update event

Bit	Name	Reset	Access	Description
10	CCVBV2	0	R	**CC2 CCVB Valid**

This field indicates that the TIMERn_CC2_CCVB registers contain data which have not been written to TIMERn_CC2_CCV. These bits are only used in output compare/PWM mode and are cleared when CCMODE is written to 0b00 (Off).

Value	Description
0	TIMERn_CC2_CCVB does not contain valid data
1	TIMERn_CC2_CCVB contains valid data which will be written to TIMERn_CC2_CCV on the next update event

Bit	Name	Reset	Access	Description
9	CCVBV1	0	R	**CC1 CCVB Valid**

This field indicates that the TIMERn_CC1_CCVB registers contain data which have not been written to TIMERn_CC1_CCV. These bits are only used in output compare/PWM mode and are cleared when CCMODE is written to 0b00 (Off).

Value	Description
0	TIMERn_CC1_CCVB does not contain valid data
1	TIMERn_CC1_CCVB contains valid data which will be written to TIMERn_CC1_CCV on the next update event

Bit	Name	Reset	Access	Description
8	CCVBV0	0	R	**CC0 CCVB Valid**

This field indicates that the TIMERn_CC0_CCVB registers contain data which have not been written to TIMERn_CC0_CCV. These bits are only used in output compare/PWM mode and are cleared when CCMODE is written to 0b00 (Off).

Value	Description
0	TIMERn_CC0_CCVB does not contain valid data
1	TIMERn_CC0_CCVB contains valid data which will be written to TIMERn_CC0_CCV on the next update event

Bit	Name	Reset	Access	Description
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	TOPBV	0	R	**TOPB 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.

Value	Description
0	TIMERn_TOPB does not contain valid data
1	TIMERn_TOPB contains valid data which will be written to TIMERn_TOP on the next update event

Bit	Name	Reset	Access	Description
1	DIR	0	R	**Direction**

Indicates count direction.

Value	Mode	Description
0	UP	Counting up
1	DOWN	Counting down

Bit	Name	Reset	Access	Description
0	RUNNING	0	R	**Running**
	Indicates if timer is running or not.			

22.5.4 TIMERn_IF - Interrupt Flag 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																					0	0	0	0	0	0	0	0		0	0	0
Access																					R	R	R	R	R	R	R	R		R	R	R
Name																					ICBOF3	ICBOF2	ICBOF1	ICBOF0	CC3	CC2	CC1	CC0		DIRCHG	UF	OF

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	ICBOF3	0	R	**CC Channel 3 Input Capture Buffer Overflow Interrupt Flag**
	This bit indicates that a new capture value has pushed an unread value out of TIMERn_CC3_CCVB.			
10	ICBOF2	0	R	**CC Channel 2 Input Capture Buffer Overflow Interrupt Flag**
	This bit indicates that a new capture value has pushed an unread value out of TIMERn_CC2_CCVB.			
9	ICBOF1	0	R	**CC Channel 1 Input Capture Buffer Overflow Interrupt Flag**
	This bit indicates that a new capture value has pushed an unread value out of TIMERn_CC1_CCVB.			
8	ICBOF0	0	R	**CC Channel 0 Input Capture Buffer Overflow Interrupt Flag**
	This bit indicates that a new capture value has pushed an unread value out of TIMERn_CC0_CCVB.			
7	CC3	0	R	**CC Channel 3 Interrupt Flag**
	This bit indicates that there has been an interrupt event on Compare/Capture channel 3.			
6	CC2	0	R	**CC Channel 2 Interrupt Flag**
	This bit indicates that there has been an interrupt event on Compare/Capture channel 2.			
5	CC1	0	R	**CC Channel 1 Interrupt Flag**
	This bit indicates that there has been an interrupt event on Compare/Capture channel 1.			
4	CC0	0	R	**CC Channel 0 Interrupt Flag**
	This bit indicates that there has been an interrupt event on Compare/Capture channel 0.			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DIRCHG	0	R	**Direction Change Detect Interrupt Flag**
	This bit is set when count direction changes. Set only in Quadrature Decoder mode			
1	UF	0	R	**Underflow Interrupt Flag**
	This bit indicates that there has been an underflow.			
0	OF	0	R	**Overflow Interrupt Flag**
	This bit indicates that there has been an overflow.			

22.5.5 TIMERn_IFS - Interrupt Flag 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																					0	0	0	0	0	0	0	0		0	0	0
Access																					W1	W1	W1	W1	W1	W1	W1	W1		W1	W1	W1
Name																					ICBOF3	ICBOF2	ICBOF1	ICBOF0	CC3	CC2	CC1	CC0		DIRCHG	UF	OF

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	ICBOF3	0	W1	**Set ICBOF3 Interrupt Flag**
	Write 1 to set the ICBOF3 interrupt flag			
10	ICBOF2	0	W1	**Set ICBOF2 Interrupt Flag**
	Write 1 to set the ICBOF2 interrupt flag			
9	ICBOF1	0	W1	**Set ICBOF1 Interrupt Flag**
	Write 1 to set the ICBOF1 interrupt flag			
8	ICBOF0	0	W1	**Set ICBOF0 Interrupt Flag**
	Write 1 to set the ICBOF0 interrupt flag			
7	CC3	0	W1	**Set CC3 Interrupt Flag**
	Write 1 to set the CC3 interrupt flag			
6	CC2	0	W1	**Set CC2 Interrupt Flag**
	Write 1 to set the CC2 interrupt flag			
5	CC1	0	W1	**Set CC1 Interrupt Flag**
	Write 1 to set the CC1 interrupt flag			
4	CC0	0	W1	**Set CC0 Interrupt Flag**
	Write 1 to set the CC0 interrupt flag			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DIRCHG	0	W1	**Set DIRCHG Interrupt Flag**
	Write 1 to set the DIRCHG interrupt flag			
1	UF	0	W1	**Set UF Interrupt Flag**
	Write 1 to set the UF interrupt flag			
0	OF	0	W1	**Set OF Interrupt Flag**
	Write 1 to set the OF interrupt flag			

22.5.6 TIMERn_IFC - Interrupt Flag 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																					0	0	0	0	0	0	0	0		0	0	0
Access																					(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1		(R)W1	(R)W1	(R)W1
Name																					ICBOF3	ICBOF2	ICBOF1	ICBOF0	CC3	CC2	CC1	CC0		DIRCHG	UF	OF

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	ICBOF3	0	(R)W1	**Clear ICBOF3 Interrupt Flag**
	Write 1 to clear the ICBOF3 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
10	ICBOF2	0	(R)W1	**Clear ICBOF2 Interrupt Flag**
	Write 1 to clear the ICBOF2 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	ICBOF1	0	(R)W1	**Clear ICBOF1 Interrupt Flag**
	Write 1 to clear the ICBOF1 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	ICBOF0	0	(R)W1	**Clear ICBOF0 Interrupt Flag**
	Write 1 to clear the ICBOF0 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	CC3	0	(R)W1	**Clear CC3 Interrupt Flag**
	Write 1 to clear the CC3 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
6	CC2	0	(R)W1	**Clear CC2 Interrupt Flag**
	Write 1 to clear the CC2 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
5	CC1	0	(R)W1	**Clear CC1 Interrupt Flag**
	Write 1 to clear the CC1 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
4	CC0	0	(R)W1	**Clear CC0 Interrupt Flag**
	Write 1 to clear the CC0 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DIRCHG	0	(R)W1	**Clear DIRCHG Interrupt Flag**
	Write 1 to clear the DIRCHG interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
1	UF	0	(R)W1	**Clear UF Interrupt Flag**
	Write 1 to clear the UF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	OF	0	(R)W1	**Clear OF Interrupt Flag**
	Write 1 to clear the OF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

22.5.7 TIMERn_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																					0	0	0	0	0	0	0	0		0	0	0
Access																					RW	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW
Name																					ICBOF3	ICBOF2	ICBOF1	ICBOF0	CC3	CC2	CC1	CC0		DIRCHG	UF	OF

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	ICBOF3	0	RW	**ICBOF3 Interrupt Enable**
	Enable/disable the ICBOF3 interrupt			
10	ICBOF2	0	RW	**ICBOF2 Interrupt Enable**
	Enable/disable the ICBOF2 interrupt			
9	ICBOF1	0	RW	**ICBOF1 Interrupt Enable**
	Enable/disable the ICBOF1 interrupt			
8	ICBOF0	0	RW	**ICBOF0 Interrupt Enable**
	Enable/disable the ICBOF0 interrupt			
7	CC3	0	RW	**CC3 Interrupt Enable**
	Enable/disable the CC3 interrupt			
6	CC2	0	RW	**CC2 Interrupt Enable**
	Enable/disable the CC2 interrupt			
5	CC1	0	RW	**CC1 Interrupt Enable**
	Enable/disable the CC1 interrupt			
4	CC0	0	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	0	RW	**DIRCHG Interrupt Enable**
	Enable/disable the DIRCHG interrupt			
1	UF	0	RW	**UF Interrupt Enable**
	Enable/disable the UF interrupt			
0	OF	0	RW	**OF Interrupt Enable**
	Enable/disable the OF interrupt			

22.5.8 TIMERn_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	0x0000FFFF																															
Access	RWH																															
Name	TOP																															

Bit	Name	Reset	Access	Description
31:0	TOP	0x0000FFFF	RWH	**Counter Top Value**
	These bits hold the TOP value for the counter.			

22.5.9 TIMERn_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	0x00000000																															
Access	RW																															
Name	TOPB																															

Bit	Name	Reset	Access	Description
31:0	TOPB	0x00000000	RW	**Counter Top Value Buffer**
	These bits hold the TOP buffer value.			

22.5.10 TIMERn_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	0x00000000																															
Access	RWH																															
Name	CNT																															

Bit	Name	Reset	Access	Description
31:0	CNT	0x00000000	RWH	**Counter Value**
	These bits hold the counter value.			

22.5.11 TIMERn_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																	0x0000															
Access																	RWH															
Name																	TIMERLOCKKEY															

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	TIMERLOCKKEY	0x0000	RWH	**Timer Lock Key**

Write any other value than the unlock code to lock TIMERn_CTRL, TIMERn_CMD, TIMERn_TOP, TIMERn_CNT, TIMERn_CCx_CTRL and TIMERn_CCx_CCV from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.

Mode	Value	Description
Read Operation		
UNLOCKED	0	TIMER registers are unlocked
LOCKED	1	TIMER registers are locked
Write Operation		
LOCK	0	Lock TIMER registers
UNLOCK	0xCE80	Unlock TIMER registers

22.5.12 TIMERn_ROUTEPEN - I/O Routing Pin Enable 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																						0	0	0					0	0	0	0
Access																						RW	RW	RW					RW	RW	RW	RW
Name																						CDTI2PEN	CDTI1PEN	CDTI0PEN					CC3PEN	CC2PEN	CC1PEN	CC0PEN

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	CDTI2PEN	0	RW	**CC Channel 2 Complementary Dead-Time Insertion Pin Enable**
	Enable/disable CC channel 2 complementary dead-time insertion output connection to pin.			
9	CDTI1PEN	0	RW	**CC Channel 1 Complementary Dead-Time Insertion Pin Enable**
	Enable/disable CC channel 1 complementary dead-time insertion output connection to pin.			
8	CDTI0PEN	0	RW	**CC Channel 0 Complementary Dead-Time Insertion Pin Enable**
	Enable/disable CC channel 0 complementary dead-time insertion output connection to pin.			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	CC3PEN	0	RW	**CC Channel 3 Pin Enable**
	Enable/disable CC channel 3 output/input connection to pin.			
2	CC2PEN	0	RW	**CC Channel 2 Pin Enable**
	Enable/disable CC channel 2 output/input connection to pin.			
1	CC1PEN	0	RW	**CC Channel 1 Pin Enable**
	Enable/disable CC channel 1 output/input connection to pin.			
0	CC0PEN	0	RW	**CC Channel 0 Pin Enable**
	Enable/disable CC Channel 0 output/input connection to pin.			

22.5.13 TIMERn_ROUTELOC0 - I/O Routing Location 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			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			CC3LOC								CC2LOC								CC1LOC								CC0LOC					

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	CC3LOC	0x00	RW	**I/O Location**

Decides the location of the CC3 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6

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*		
21:16	CC2LOC	0x00	RW	**I/O Location**

Decides the location of the CC2 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

Bit	Name	Reset	Access	Description
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
13:8	CC1LOC	0x00	RW	**I/O Location**

Decides the location of the CC1 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	CC0LOC	0x00	RW	**I/O Location**

Decides the location of the CC0 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

22.5.14 TIMERn_ROUTELOC2 - I/O Routing Location 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											0x00								0x00								0x00					
Access											RW								RW								RW					
Name											CDTI2LOC								CDTI1LOC								CDTI0LOC					

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	CDTI2LOC	0x00	RW	**I/O Location**

Decides the location of the CDTI2 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4

15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CDTI1LOC	0x00	RW	**I/O Location**

Decides the location of the CDTI1 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	CDTI0LOC	0x00	RW	**I/O Location**

Decides the location of the CDTI0 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

Bit	Name	Reset	Access	Description
	3	LOC3		Location 3
	4	LOC4		Location 4

22.5.15 TIMERn_CCx_CTRL - CC Channel Control 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		0	0	0	0x0		0x0						0x0						0x0		0x0		0x0					0		0	0x0	
Access		RW	RW	RW	RW		RW						RW						RW		RW		RW					RW		RW	RW	
Name		FILT	INSEL	PRSCONF	ICEVCTRL		ICEDGE						PRSSEL						CUFOA		COFOA		CMOA					COIST		OUTINV	MODE	

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	FILT	0	RW	**Digital Filter**

Enable digital filter.

Value	Mode	Description
0	DISABLE	Digital filter disabled
1	ENABLE	Digital filter enabled

Bit	Name	Reset	Access	Description
29	INSEL	0	RW	**Input Selection**

Select Compare/Capture channel input.

Value	Mode	Description
0	PIN	TIMERnCCx pin is selected
1	PRS	PRS input (selected by PRSSEL) is selected

Bit	Name	Reset	Access	Description
28	PRSCONF	0	RW	**PRS Configuration**

Select PRS pulse or level.

Value	Mode	Description
0	PULSE	Each CC event will generate a one HFPERCLK cycle high pulse
1	LEVEL	The PRS channel will follow CC out

Bit	Name	Reset	Access	Description
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)

Bit	Name	Reset	Access	Description
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

23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	PRSSEL	0x0	RW	**Compare/Capture Channel PRS Input Channel Selection**

Select PRS input channel for Compare/Capture channel.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input
8	PRSCH8	PRS Channel 8 selected as input
9	PRSCH9	PRS Channel 9 selected as input
10	PRSCH10	PRS Channel 10 selected as input
11	PRSCH11	PRS Channel 11 selected as input
12	PRSCH12	PRS Channel 12 selected as input
13	PRSCH13	PRS Channel 13 selected as input
14	PRSCH14	PRS Channel 14 selected as input
15	PRSCH15	PRS Channel 15 selected as input

15:14	*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

Bit	Name	Reset	Access	Description
	3	SET		Set output on counter underflow
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
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	COIST	0	RW	**Compare Output Initial State**
	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	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	OUTINV	0	RW	**Output Invert**
	Setting this bit inverts the output from the CC channel (Output compare,PWM).			
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

22.5.16 TIMERn_CCx_CCV - CC Channel Value Register (Actionable Reads)

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	0x00000000																															
Access	RWH																															
Name	CCV																															

Bit	Name	Reset	Access	Description
31:0	CCV	0x00000000	RWH	**CC Channel Value**
	In input capture mode, this field holds the first unread capture value. When reading this register in input capture mode, the contents of the TIMERn_CCx_CCVB register will be written to TIMERn_CCx_CCV in the next cycle. In compare mode, this fields holds the compare value.			

22.5.17 TIMERn_CCx_CCVP - CC Channel Value Peek 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	0x00000000																															
Access	R																															
Name	CCVP																															

Bit	Name	Reset	Access	Description
31:0	CCVP	0x00000000	R	**CC Channel Value Peek**
	This field is used to read the CC value without pulling data through the FIFO in capture mode.			

22.5.18 TIMERn_CCx_CCVB - CC Channel Buffer Register

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	0x00000000																															
Access	RWH																															
Name	CCVB																															

Bit	Name	Reset	Access	Description
31:0	CCVB	0x00000000	RWH	**CC Channel Value Buffer**
	In Input Capture mode, this field holds the last capture value if the TIMERn_CCx_CCV register already contains an earlier unread capture value. In Output Compare or PWM mode, this field holds the CC buffer value which will be written to TIMERn_CCx_CCV on an update event if TIMERn_CCx_CCVB contains valid data.			

22.5.19 TIMERn_DTCTRL - DTI Control Register

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								0														0	0		0x0				0	0	0	0
Access								RW														RW	RW		RW				RW	RW	RW	RW
Name								DTPRSEN														DTFATS	DTAR		DTPRSSEL				DTCINV	DTIPOL	DTDAS	DTEN

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	DTPRSEN	0	RW	**DTI PRS Source Enable**
	Enable/disable PRS as DTI input.			
23:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	DTFATS	0	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 channel 0 to output the incoming PRS input while the other channels go to safe state.			
9	DTAR	0	RW	**DTI Always Run**
	This is used only for DTI channel 0. It Allows DTI channel 0 to keep running even when timer is stopped. This is useful when its input source is PRS. However, here the undivided HFPERCLK is always used regardless of the programmed value in DTPRESC.			
8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:4	DTPRSSEL	0x0	RW	**DTI PRS Source Channel Select**
	Selects which PRS channel compare channel 0 will listen to.			

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input
8	PRSCH8	PRS Channel 8 selected as input
9	PRSCH9	PRS Channel 9 selected as input
10	PRSCH10	PRS Channel 10 selected as input
11	PRSCH11	PRS Channel 11 selected as input

Bit	Name	Reset	Access	Description
	12	PRSCH12		PRS Channel 12 selected as input
	13	PRSCH13		PRS Channel 13 selected as input
	14	PRSCH14		PRS Channel 14 selected as input
	15	PRSCH15		PRS Channel 15 selected as input
3	DTCINV	0	RW	**DTI Complementary Output Invert**
	Set to invert complementary outputs.			
2	DTIPOL	0	RW	**DTI Inactive Polarity**
	Set inactive polarity for outputs.			
1	DTDAS	0	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	0	RW	**DTI Enable**
	Enable/disable DTI.			

22.5.20 TIMERn_DTTIME - DTI Time Control Register

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											0x00								0x00										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	0x00	RW	**DTI Fall-time**

Set time span for the falling edge.

Value	Description
DTFALLT	Fall time of DTFALLT+1 prescaled HFPERCLK cycles

Bit	Name	Reset	Access	Description
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	DTRISET	0x00	RW	**DTI Rise-time**

Set time span for the rising edge.

Value	Description
DTRISET	Rise time of DTRISET+1 prescaled HFPERCLK cycles

Bit	Name	Reset	Access	Description
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	DTPRESC	0x0	RW	**DTI Prescaler Setting**

Select prescaler for DTI.

Value	Mode	Description
0	DIV1	The HFPERCLK is undivided
1	DIV2	The HFPERCLK is divided by 2
2	DIV4	The HFPERCLK is divided by 4
3	DIV8	The HFPERCLK is divided by 8
4	DIV16	The HFPERCLK is divided by 16
5	DIV32	The HFPERCLK is divided by 32
6	DIV64	The HFPERCLK is divided by 64
7	DIV128	The HFPERCLK is divided by 128
8	DIV256	The HFPERCLK is divided by 256
9	DIV512	The HFPERCLK is divided by 512
10	DIV1024	The HFPERCLK is divided by 1024

Bit	Name	Reset	Access	Description

22.5.21 TIMERn_DTFC - DTI Fault Configuration Register

Offset	Bit Position																													
0x0A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0	0	0	0							0x0						0x0								0x0			
Access					RW	RW	RW	RW							RW						RW								RW			
Name					DTLOCKUPFEN	DTDBGFEN	DTPRS1FEN	DTPRS0FEN							DTFA						DTPRS1FSEL								DTPRS0FSEL			

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	DTLOCKUPFEN	0	RW	**DTI Lockup Fault Enable**
	Set this bit to 1 to enable core lockup as a fault source			
26	DTDBGFEN	0	RW	**DTI Debugger Fault Enable**
	Set this bit to 1 to enable debugger as a fault source			
25	DTPRS1FEN	0	RW	**DTI PRS 1 Fault Enable**
	Set this bit to 1 to enable PRS source 1(PRS channel determined by DTPRS1FSEL) as a fault source			
24	DTPRS0FEN	0	RW	**DTI PRS 0 Fault Enable**
	Set this bit to 1 to enable PRS source 0(PRS channel determined by DTPRS0FSEL) 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

15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	DTPRS1FSEL	0x0	RW	**DTI PRS Fault Source 1 Select**
	Select PRS channel for fault source 1.			

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as fault source 1
1	PRSCH1	PRS Channel 1 selected as fault source 1
2	PRSCH2	PRS Channel 2 selected as fault source 1

Bit	Name	Reset	Access	Description
	3	PRSCH3		PRS Channel 3 selected as fault source 1
	4	PRSCH4		PRS Channel 4 selected as fault source 1
	5	PRSCH5		PRS Channel 5 selected as fault source 1
	6	PRSCH6		PRS Channel 6 selected as fault source 1
	7	PRSCH7		PRS Channel 7 selected as fault source 1
	8	PRSCH8		PRS Channel 8 selected as fault source 1
	9	PRSCH9		PRS Channel 9 selected as fault source 1
	10	PRSCH10		PRS Channel 10 selected as fault source 1
	11	PRSCH11		PRS Channel 11 selected as fault source 1
	12	PRSCH12		PRS Channel 12 selected as fault source 1
	13	PRSCH13		PRS Channel 13 selected as fault source 1
	14	PRSCH14		PRS Channel 14 selected as fault source 1
	15	PRSCH15		PRS Channel 15 selected as fault source 1
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	DTPRS0FSEL	0x0	RW	**DTI PRS Fault Source 0 Select**

Select PRS channel for fault source 0.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as fault source 0
1	PRSCH1	PRS Channel 1 selected as fault source 1
2	PRSCH2	PRS Channel 2 selected as fault source 2
3	PRSCH3	PRS Channel 3 selected as fault source 3
4	PRSCH4	PRS Channel 4 selected as fault source 4
5	PRSCH5	PRS Channel 5 selected as fault source 5
6	PRSCH6	PRS Channel 6 selected as fault source 6
7	PRSCH7	PRS Channel 7 selected as fault source 7
8	PRSCH8	PRS Channel 8 selected as fault source 8
9	PRSCH9	PRS Channel 9 selected as fault source 9
10	PRSCH10	PRS Channel 10 selected as fault source 10
11	PRSCH11	PRS Channel 11 selected as fault source 11
12	PRSCH12	PRS Channel 12 selected as fault source 12
13	PRSCH13	PRS Channel 13 selected as fault source 13
14	PRSCH14	PRS Channel 14 selected as fault source 14
15	PRSCH15	PRS Channel 15 selected as fault source 15

22.5.22 TIMERn_DTOGEN - DTI Output Generation Enable Register

Offset	Bit Position																													
0x0AC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0	0	0	0	0	0
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	0	RW	**DTI CDTI2 Output Generation Enable**
	This bit enables/disables output generation for the CDTI2 output from the DTI.			
4	DTOGCDTI1EN	0	RW	**DTI CDTI1 Output Generation Enable**
	This bit enables/disables output generation for the CDTI1 output from the DTI.			
3	DTOGCDTI0EN	0	RW	**DTI CDTI0 Output Generation Enable**
	This bit enables/disables output generation for the CDTI0 output from the DTI.			
2	DTOGCC2EN	0	RW	**DTI CC2 Output Generation Enable**
	This bit enables/disables output generation for the CC2 output from the DTI.			
1	DTOGCC1EN	0	RW	**DTI CC1 Output Generation Enable**
	This bit enables/disables output generation for the CC1 output from the DTI.			
0	DTOGCC0EN	0	RW	**DTI CC0 Output Generation Enable**
	This bit enables/disables output generation for the CC0 output from the DTI.			

22.5.23 TIMERn_DTFAULT - DTI Fault Register

Offset	Bit Position																													
0x0B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0	0	0	0
Access																													R	R	R	R
Name																													DTLOCKUPF	DTDBGF	DTPRS1F	DTPRS0F

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	DTLOCKUPF	0	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	0	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	0	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	0	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.			

22.5.24 TIMERn_DTFAULTC - DTI Fault Clear Register

Offset	Bit Position																													
0x0B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													0	0	0	0
Access																													W1	W1	W1	W1
Name																													TLOCKUPFC	DTDBGFC	DTPRS1FC	DTPRS0FC

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	TLOCKUPFC	0	W1	**DTI Lockup Fault Clear**
	Write 1 to this bit to clear core lockup fault.			
2	DTDBGFC	0	W1	**DTI Debugger Fault Clear**
	Write 1 to this bit to clear debugger fault.			
1	DTPRS1FC	0	W1	**DTI PRS1 Fault Clear**
	Write 1 to this bit to clear PRS 1 fault.			
0	DTPRS0FC	0	W1	**DTI PRS0 Fault Clear**
	Write 1 to this bit to clear PRS 0 fault.			

22.5.25 TIMERn_DTLOCK - DTI Configuration Lock Register

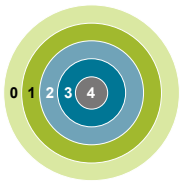
Offset	Bit Position																													
0x0B8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0000															
Access																	RWH															
Name																	LOCKKEY															

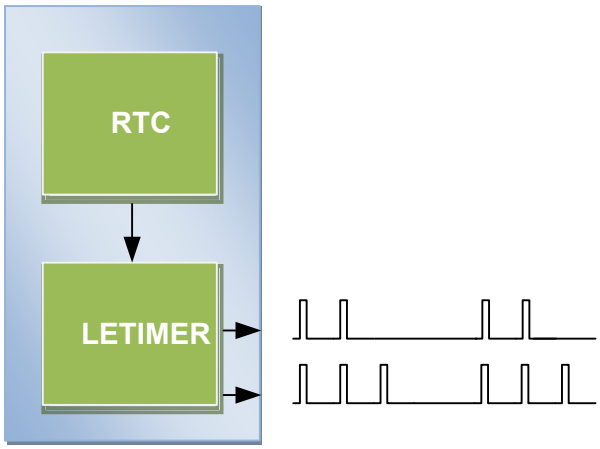
Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x0000	RWH	**DTI Lock Key**

Write any other value than the unlock code to lock TIMERn_ROUTE, TIMERn_DTCTRL, TIMERn_DTTIME and TIMERn_DTFC from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.

Mode	Value	Description
Read Operation		
UNLOCKED	0	TIMER DTI registers are unlocked
LOCKED	1	TIMER DTI registers are locked
Write Operation		
LOCK	0	Lock TIMER DTI registers
UNLOCK	0xCE80	Unlock TIMER DTI registers

23. 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 RTC.

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 RTC using PRS, allowing advanced time-keeping and wake-up functions in EM2 DeepSleep and EM3 Stop

23.1 Introduction

The unique LETIMER™, the Low Energy Timer, is a 16-bit timer that is available in energy mode EM0 Active, EM1 Sleep, EM2 DeepSleep, and EM3 Stop. 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.

The LETIMER can be used to output a variety of waveforms with minimal software intervention. It can also be connected to the Real Time Counter (RTC) using PRS, and can be configured to start counting on compare matches from the RTC.

23.2 Features

- 16-bit down count timer
- 2 Compare match registers
- Compare register 0 can be top timer top value
- Compare registers can be double buffered
- Double buffered 8-bit Repeat Register
- Same clock source as the Real Time Counter
- LETIMER can be triggered (started) by an RTC event via PRS or by software
- LETIMER can be started, stopped, and/or cleared by PRS
- 2 output pins can optionally be configured to provide different waveforms on timer underflow:
  - Toggle output pin
  - Apply a positive pulse (pulse width of one $LFACLK_{LETIMER}$ period)
  - PWM
- Interrupt on:
  - Compare matches
  - Timer underflow
  - Repeat done
- Optionally runs during debug
- PRS Output

23.3 Functional Description

An overview of the LETIMER module is shown in Figure 23.1 LETIMER Overview on page 883. The LETIMER is a 16-bit down-counter with two compare registers, LETIMERn_COMP0 and LETIMERn_COMP1. The LETIMERn_COMP0 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_COMP0 and LETIMERn_REP0 registers can be double buffered by the LETIMERn_COMP1 and LETIMERn_REP1 registers to allow continuous operation. The timer can generate a single pin output, or two linked outputs.

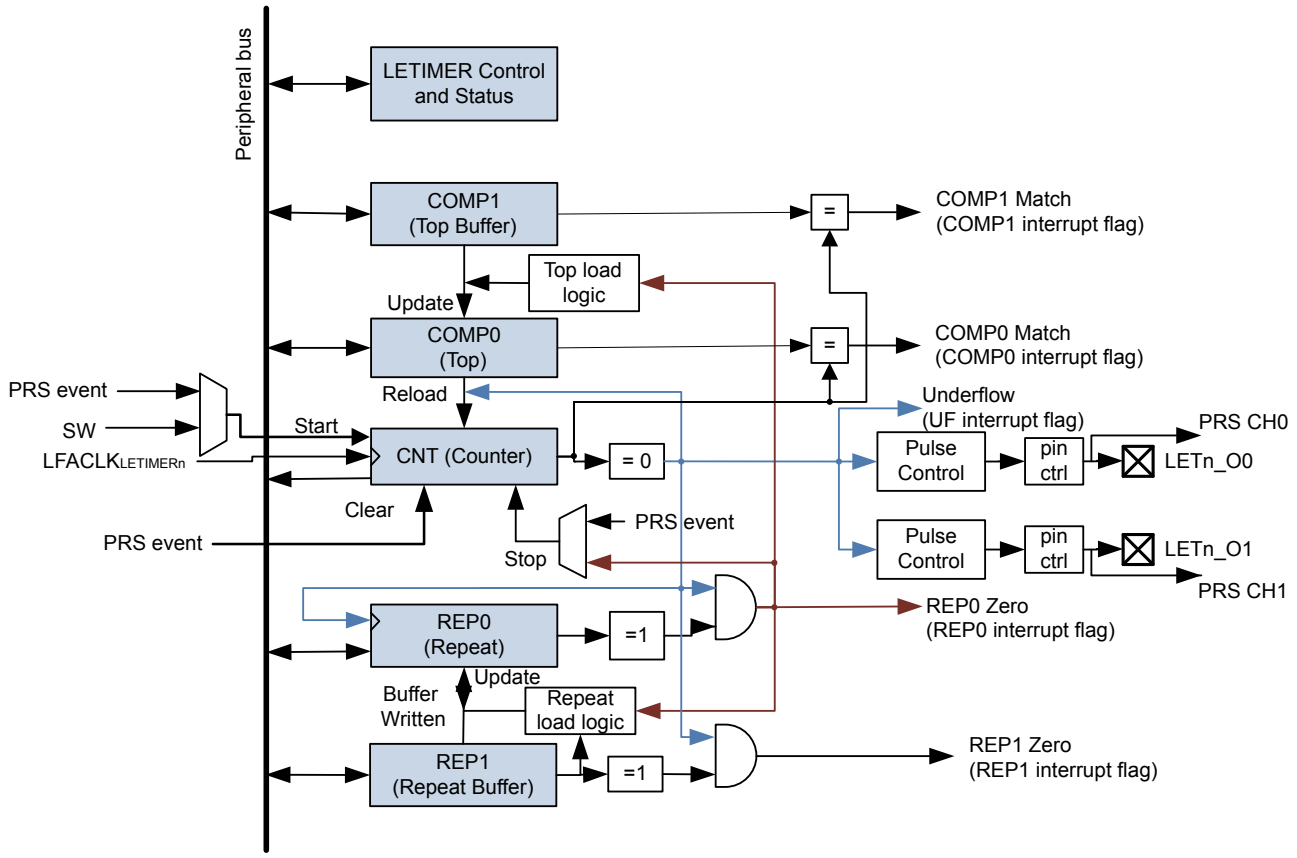


Figure 23.1. LETIMER Overview

23.3.1 Timer

The timer is started by setting command bit START in LETIMERn_CMD, and stopped by setting the STOP command bit in the same register. RUNNING in LETIMERn_STATUS is set as long as the timer is running. The timer can also be started on external signals, such as a compare match from the Real Time Counter. If START and STOP are set at the same time, STOP has priority, and the timer will be stopped.

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.

23.3.2 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 becomes equal to their value. When LETIMERn_CNT becomes equal to the value of LETIMERn_COMP0, the interrupt flag COMP0 in LETIMERn_IF is set, and when LETIMERn_CNT becomes equal to the value of LETIMERn_COMP1, the interrupt flag COMP1 in LETIMERn_IF is set.

23.3.3 Top Value

If COMP0TOP in LETIMERn_CTRL is set, the value of LETIMERn_COMP0 acts as the top value of the timer, and LETIMERn_COMP0 is loaded into LETIMERn_CNT on timer underflow. If COMP0TOP is cleared to 0, the timer wraps around to 0xFFFF. The underflow interrupt flag UF in LETIMERn_IF is set when the timer reaches zero.

23.3.3.1 Buffered Top Value

If BUFTOP in LETIMERn_CTRL is set, the value of LETIMERn_COMP0 is buffered by LETIMERn_COMP1. In this mode, the value of LETIMERn_COMP1 is loaded into LETIMERn_COMP0 every time LETIMERn_REP0 is about to decrement to 0. This can for instance be used in conjunction with the buffered repeat mode to generate continually changing output waveforms.

Write operations to LETIMERn_COMP0 have priority over buffer loads.

23.3.3.2 Repeat Modes

By default, the timer wraps around to the top value or 0xFFFF 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 23.1 LETIMER Repeat Modes on page 884.

Table 23.1. 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, it is loaded into LETIMERn_REP0 when LETIMERn_REP0 is about to be decremented to 0.
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.

23.3.3.3 Free-Running Mode

In free-running mode, the LETIMER acts as a regular timer and the repeat counter is disabled. When started, the timer runs until it is stopped using the STOP command bit in LETIMERn_CMD. A state machine for this mode is shown in Figure 23.2 LETIMER State Machine for Free-running Mode on page 885 .

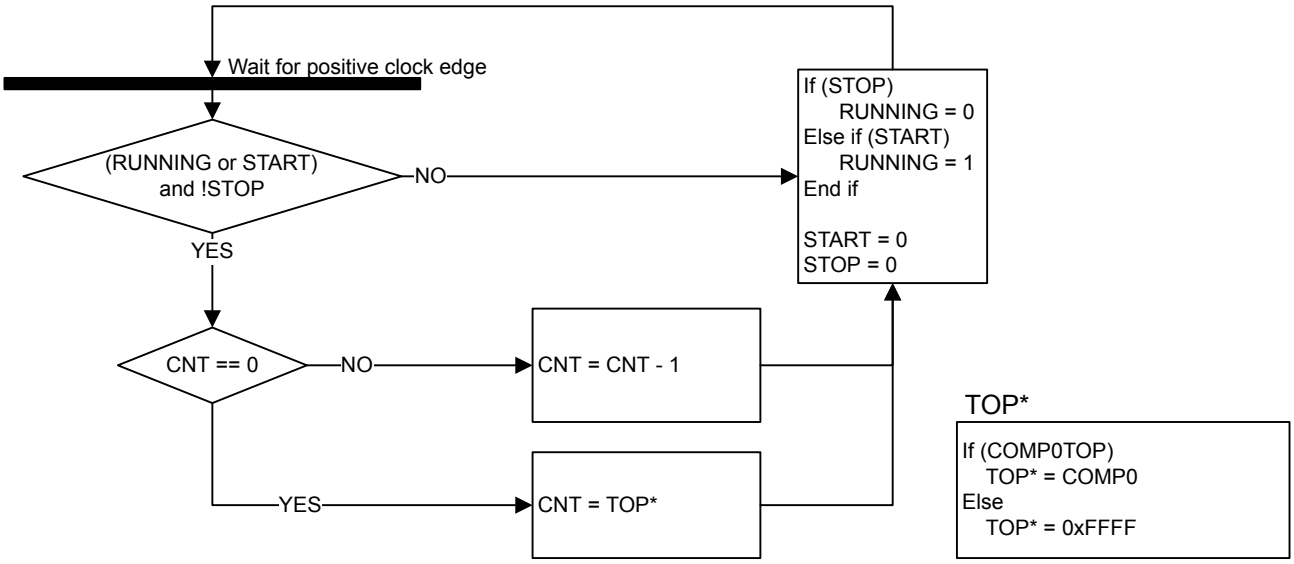


Figure 23.2. LETIMER State Machine for Free-running Mode

Note that the CLEAR command bit in LETIMERn_CMD always has priority over other changes 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 0xFFFF. Since no underflow event is generated, no output action is performed. LETIMERn_REP0, LETIMERn_REP1, LETIMERn_COMP0 and LETIMERn_COMP1 are also left untouched.

23.3.3.4 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: 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. Figure 23.3 LETIMER One-shot Repeat State Machine on page 886 .

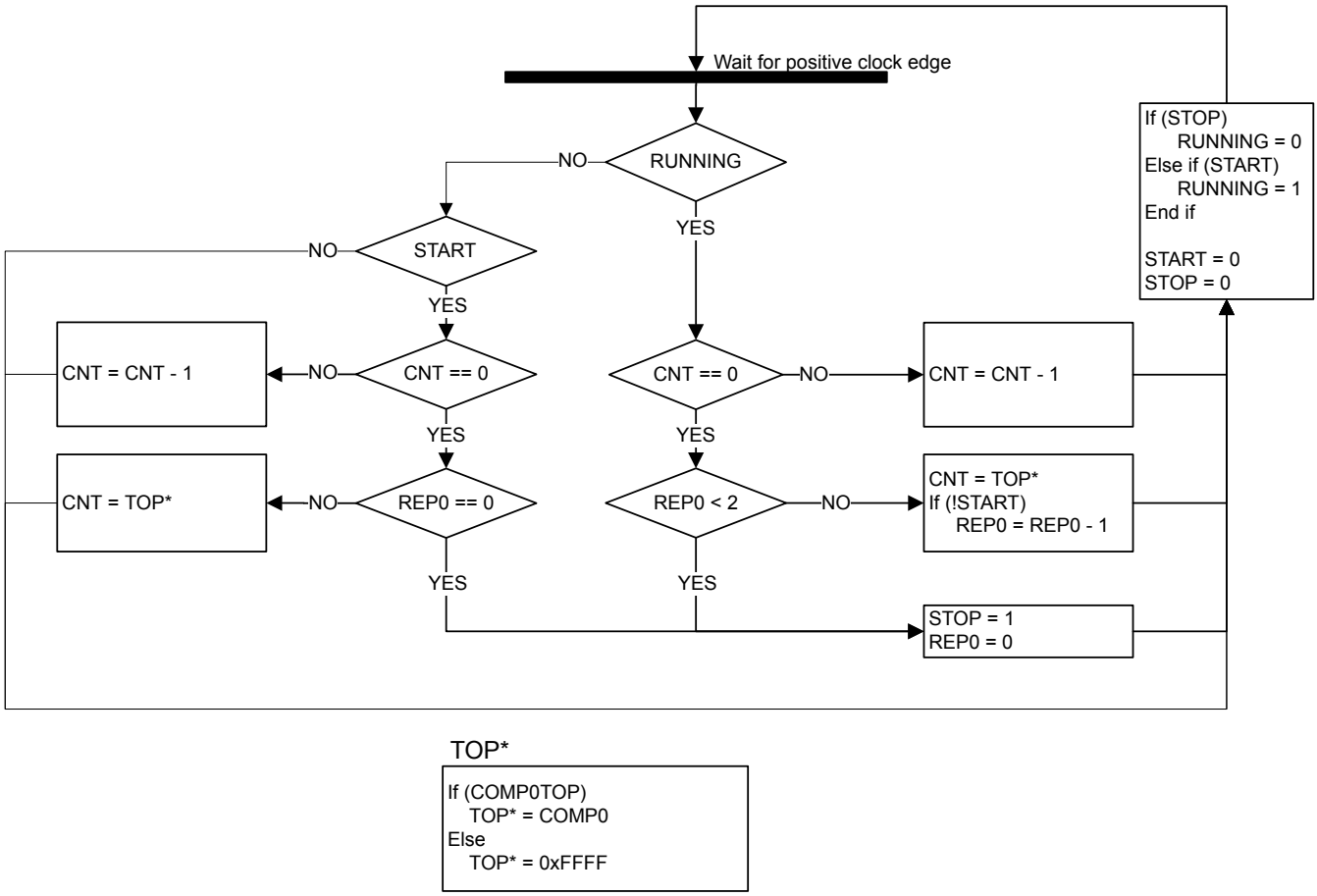


Figure 23.3. LETIMER One-shot Repeat State Machine

23.3.3.5 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 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_COMP0) may also optionally be buffered 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 23.4 LETIMER Buffered Repeat State Machine on page 887. $REP1_{USED}$ 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 written.

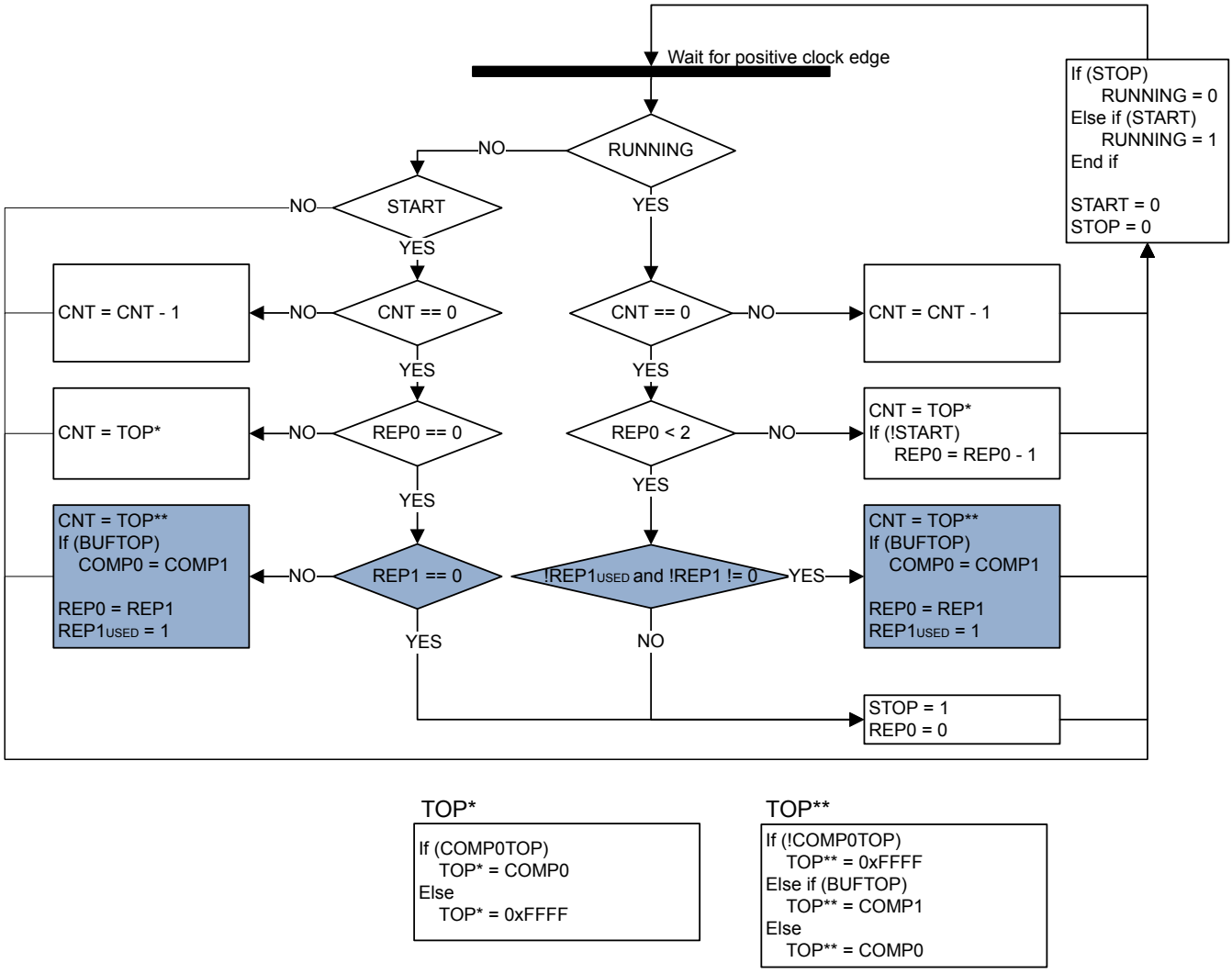


Figure 23.4. LETIMER Buffered Repeat State Machine

23.3.3.6 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 23.5 LETIMER Double Repeat State Machine on page 888.

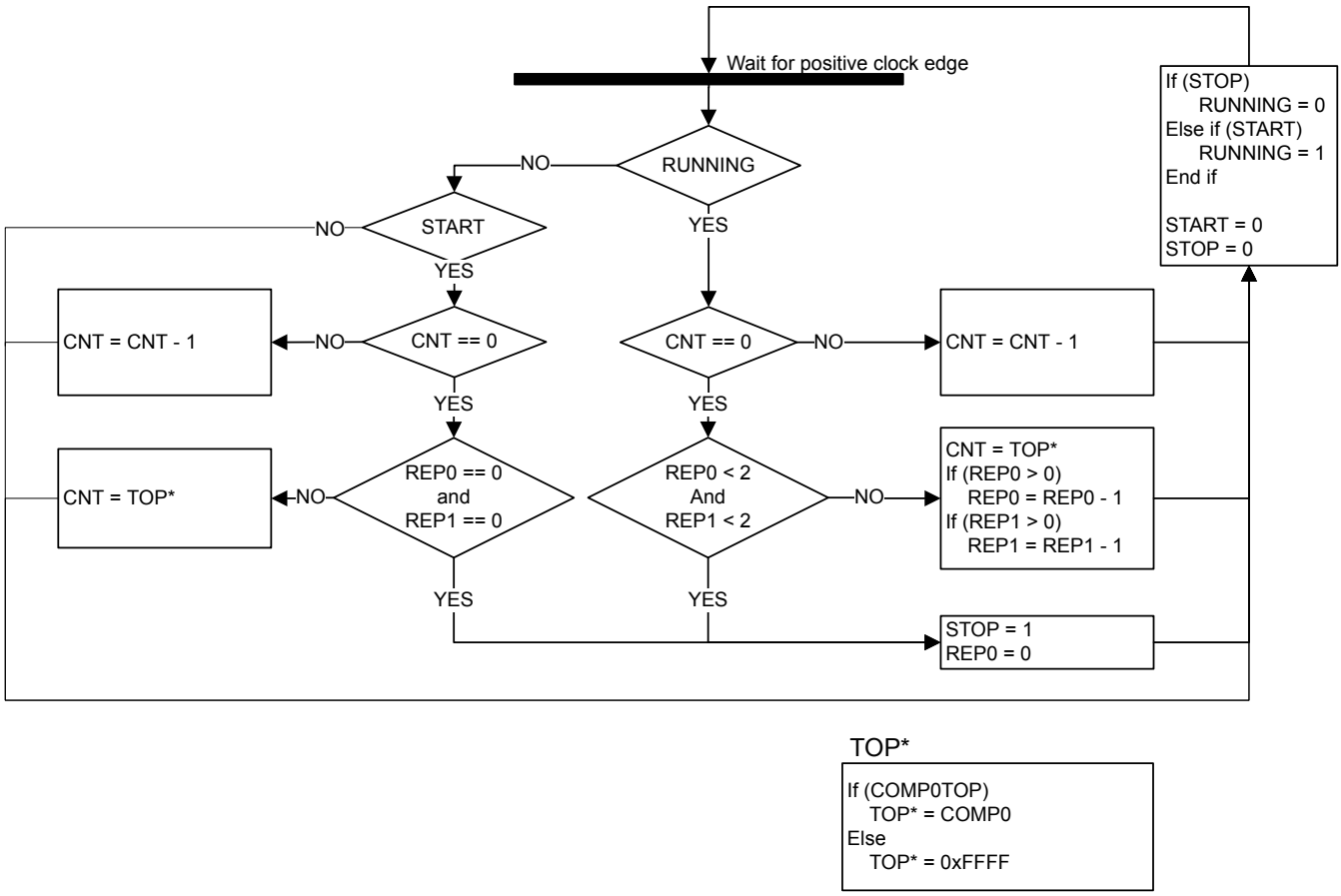


Figure 23.5. LETIMER Double Repeat State Machine

23.3.3.7 Clock Source

The LETIMER clock source and its prescaler value are defined in the Clock Management Unit (CMU). The $LFACLK_{LETIMERn}$ has a frequency given by Figure 23.6 LETIMER Clock Frequency on page 888.

$$f_{LFACKL\_LETIMERn} = 32768/2^{LETIMERn}$$

Figure 23.6. LETIMER Clock Frequency

where the exponent LETIMERn is a 4 bit value in the CMU_LFAPRESC0 register.

To use this module, the LE interface clock must be enabled in CMU_HFBUSCLKEN0, in addition to the module clock.

23.3.3.8 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 23.7 LETIMER PRS Input Triggers on page 889.

There are 12 PRS inputs to the LETIMER. PRSSTARTSEL, PRSSTOPSEL, and PRSCLEARSEL select which PRS inputs are used to start, stop, and/or clear the LETIMER. PRSSTARTMODE, PRSSTOPMODE, and PRSCLEARMODE select which edge or edge(s) can trigger the start, stop, and/or clear action.The PRSSTARTEN, PRSSTOPEN, and PRSCLEAREN signals shown in the diagram are derived from the PRSSTARTMODE, PRSSTOPMODE, and PRSCLEARMODE fields; if the corresponding bit field is set to NONE, the feature is disabled.

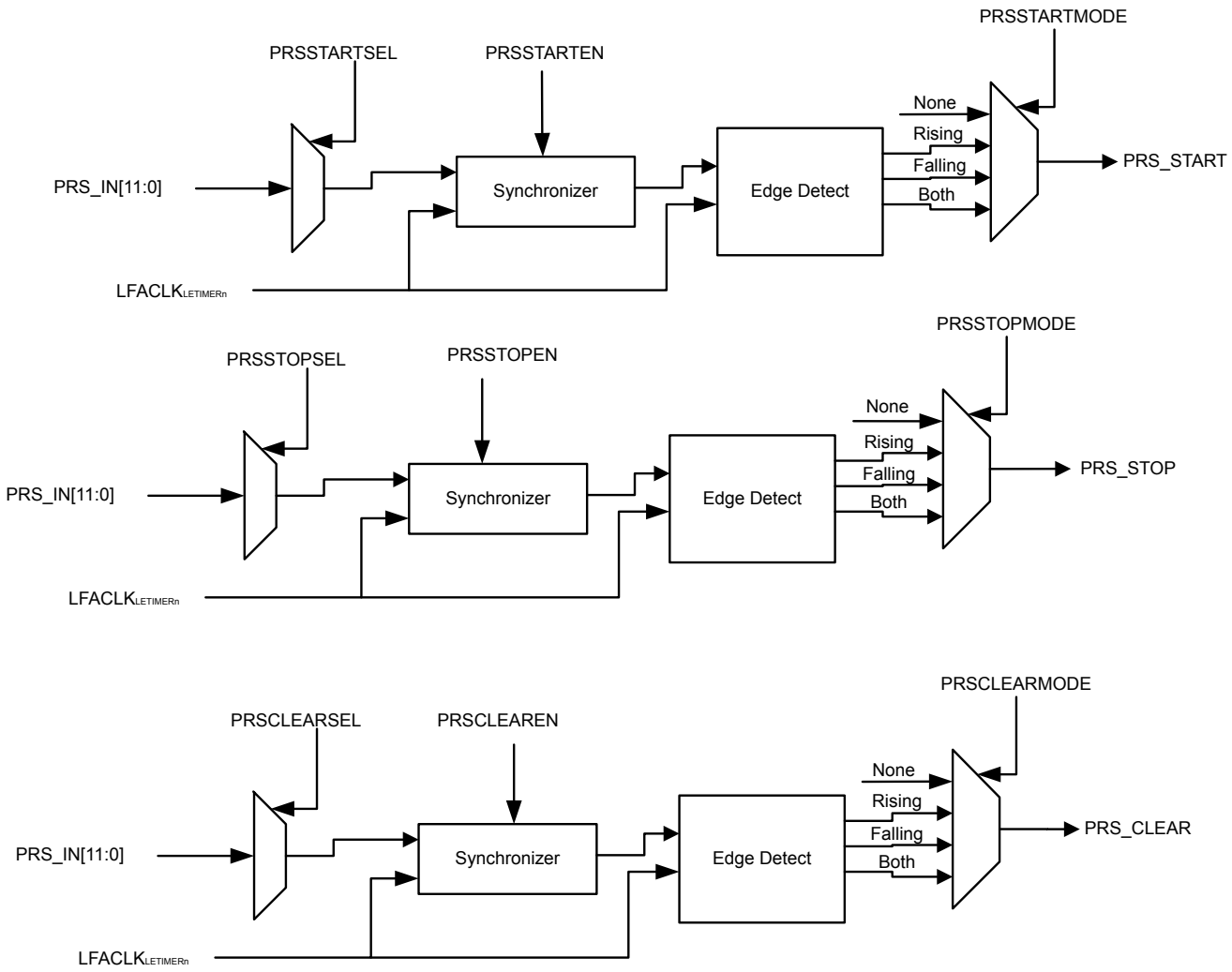


Figure 23.7. LETIMER PRS Input Triggers

23.3.3.9 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.

23.3.4 Underflow Output Action

For each of the repeat registers, an underflow output action can be set. The configured output action is performed every time the counter underflows while the respective repeat register is nonzero. In PWM mode, the output is similarly only changed on COMP1 match if the repeat register is nonzero. As an example, the timer will perform 7 output actions if LETIMERn_REP0 is set to 7 when starting the timer in one-shot mode and leaving it untouched.

The output actions can be set by configuring UFOA0 and UFOA1 in LETIMERn_CTRL. UFOA0 defines the action on output 0, and is connected to LETIMERn_REP0, while UFOA1 defines the action on output 1 and is connected to LETIMERn_REP1. The possible actions are defined in Table 23.2 LETIMER Underflow Output Actions on page 890.

Table 23.2. 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 if LEIMERn_REPx is nonzero
0b10	Pulse	The output is held active for one clock cycle on LETIMERn_CNT underflow if LETIMERn_REPx is nonzero. 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_COMP1 if LETIMERn_REPx is nonzero.

Note:

- For the Pulse and PWM modes, the outputs will return to their idle states regardless of the state of the corresponding LETIMERn_REPx registers. They will only be set active if the LETIMERn_REPx registers are nonzero however.
- For free-running mode, LETIMERn_REP0 != 0 for output generation to be enabled.

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.

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.

Some simple waveforms generated with the different output modes are shown in Figure 23.8 LETIMER Simple Waveforms Output on page 891. For the example, REPMODE in LETIMERn_CTRL has been cleared, COMP0TOP also in LETIMERn_CTRL has been set and LETIMERn_COMP0 has been written to 3. As seen in the figure, LETIMERn_COMP0 now decides the length of the signal periods. For the toggle mode, the period of the output signal is 2(LETIMERn_COMP0 + 1), and for the pulse modes, the periods of the output signals are LETIMERn_COMP0+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.

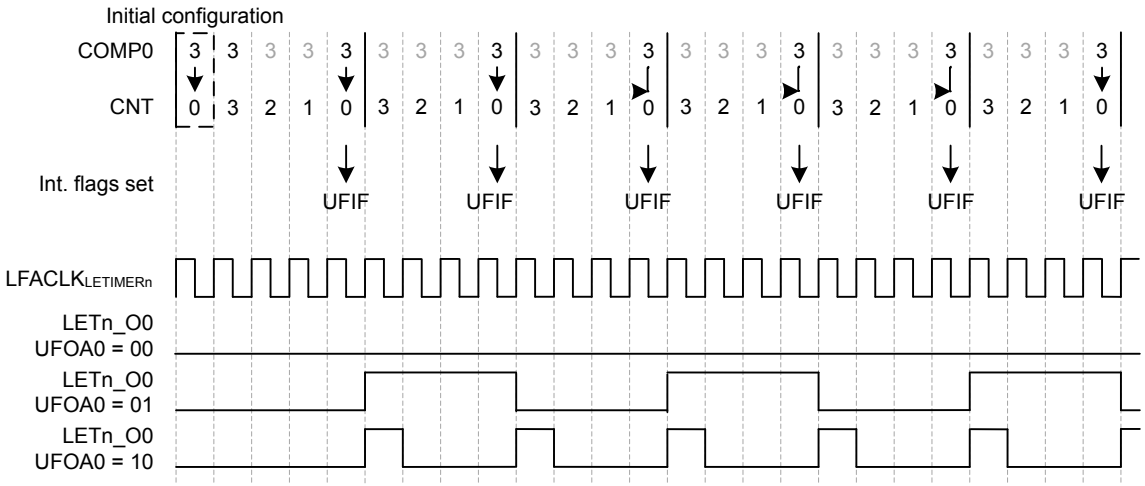


Figure 23.8. LETIMER Simple Waveforms Output

For the example in Figure 23.9 LETIMER Repeated Counting on page 891, the One-shot repeat mode has been selected, 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.

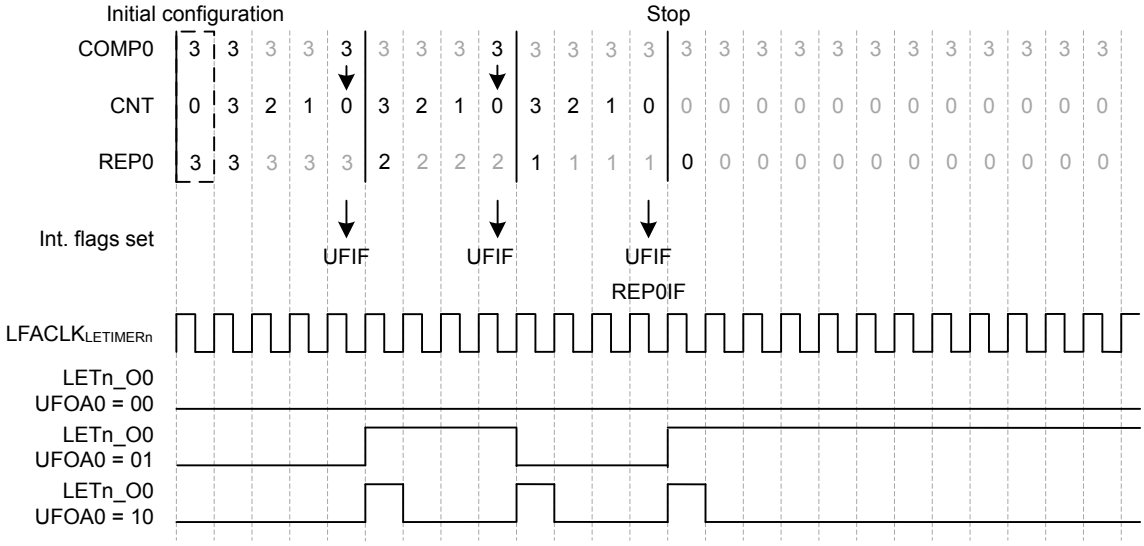


Figure 23.9. LETIMER Repeated Counting

Using the Double repeat mode, output can be generated on both the LETIMER outputs. Figure 23.10 LETIMER Dual Output on page 892 shows an example of this. 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.

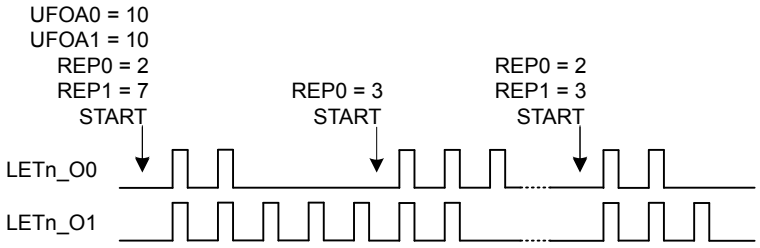


Figure 23.10. LETIMER Dual Output

23.3.5 PRS Output

The LETIMER outputs can be routed out onto the PRS system. Enabling the PRS connection can be done by setting SOURCESEL to LETIMERx and SIGSEL to LETIMERxCHn in PRS_CHx_CTRL. The PRS register description can be found in 15.5 Register Description

23.3.6 Examples

This section presents a couple of usage examples for the LETIMER.

23.3.6.1 Triggered Output Generation

If both LETIMERn_CNT and LETIMERn_REP0 are 0 in buffered mode, and COMP0TOP and BUFTOP in LETIMERn_CTRL are set, the values of LETIMERn_COMP1 and LETIMERn_REP1 are loaded into LETIMERn_CNT 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_COMP1 determines the period lengths.

As the RTC can be used via PRS to start the LETIMER, the RTC and LETIMER can thus be combined to generate specific pulse-trains at given intervals. Software can update LETIMERn_COMP1 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 23.11 LETIMER Triggered Operation on page 893, 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.

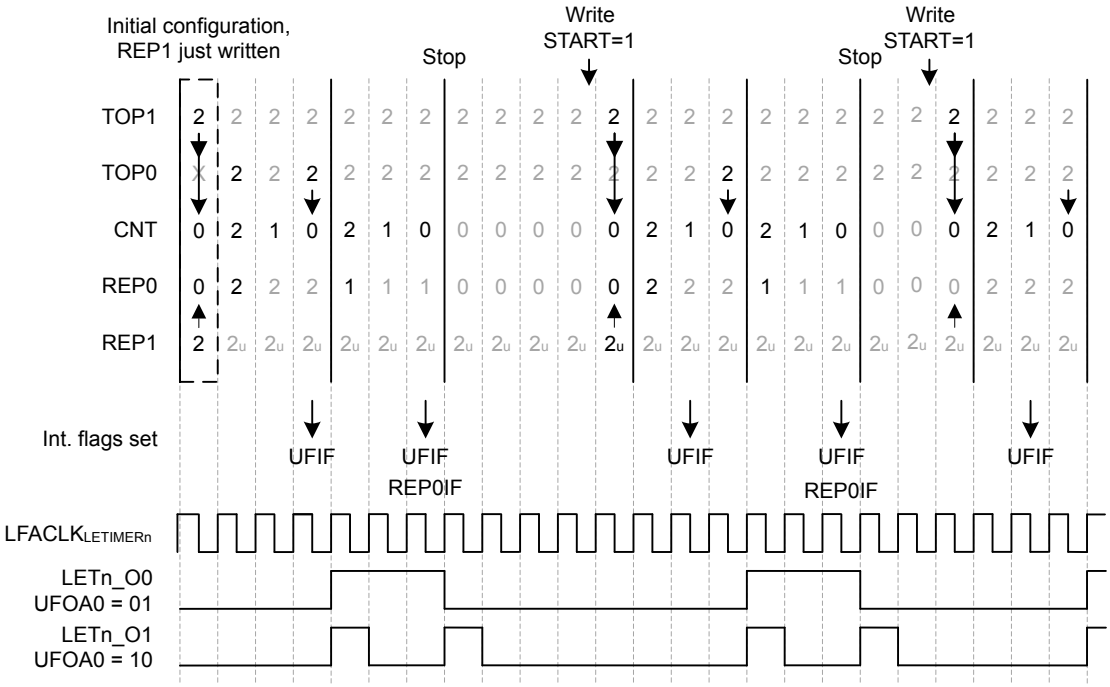


Figure 23.11. LETIMER Triggered Operation

23.3.6.2 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 23.8 LETIMER Simple Waveforms Output on page 891, but to generate changing waveforms, using the repeat counter and buffer registers can prove advantageous.

For the example in Figure 23.12 LETIMER Continuous Operation on page 894, 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

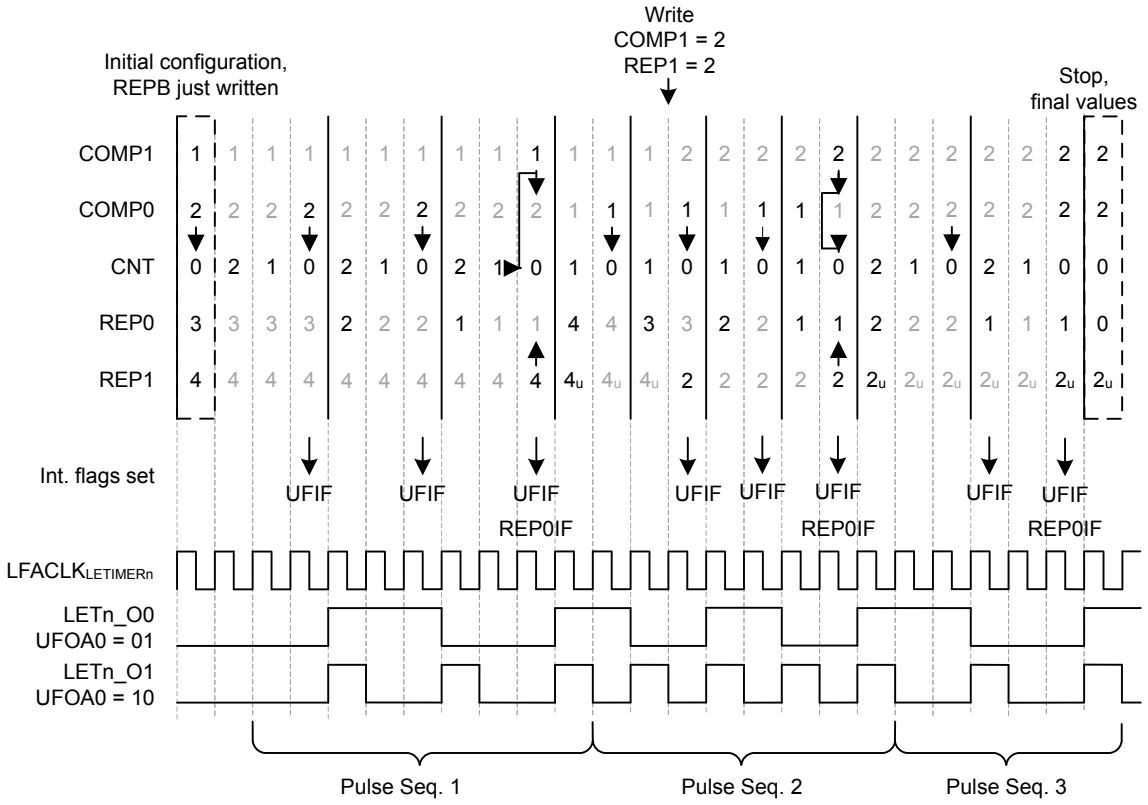


Figure 23.12. LETIMER Continuous Operation

The first two sequences are loaded into the LETIMER before the timer is started.

LETIMERn_COMP0 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. COMP1 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 23.12 LETIMER Continuous Operation on page 894. 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 23.12 LETIMER Continuous Operation on page 894 assumes that writes are done in advance so they arrive in the LETIMER as described in the figure.

Figure 23.13 LETIMER LETIMERn_CNT Not Initialized to 0 on page 895 shows 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.

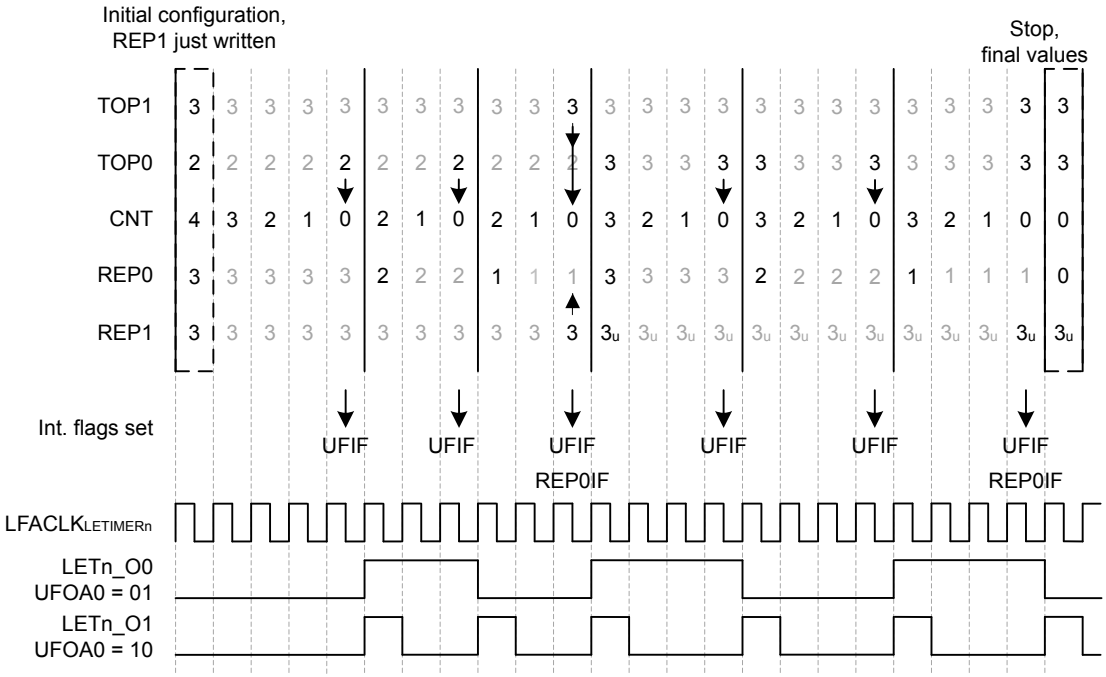


Figure 23.13. LETIMER LETIMERn_CNT Not Initialized to 0

23.3.6.3 PWM Output

There are several ways of generating PWM output with the LETIMER, but the most straight-forward way is using 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 idle on timer underflow, and active on LETIMERn_COMP1 match, so if for instance COMP0TOP = 1 and OPOL0 = 0 in LETIMERn_CTRL, LETIMERn_COMP0 determines the PWM period, and LETIMERn_COMP1 determines the active period.

The PWM period in PWM mode is LETIMERn_COMP0 + 1. There is no special handling of the case where LETIMERn_COMP1 > LETIMERn_COMP0, so if LETIMERn_COMP1 > LETIMERn_COMP0, 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_COMP1 is set to a value larger than LETIMERn_COMP0.

23.3.6.4 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.

23.3.7 Register Access

This module is a Low Energy Peripheral, and supports immediate synchronization. For description regarding immediate synchronization, the reader is referred to 4.3.1 Writing.

23.4 Register Map

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

Offset	Name	Type	Description
0x000	LETIMERn_CTRL	RW	Control Register
0x004	LETIMERn_CMD	W1	Command Register
0x008	LETIMERn_STATUS	R	Status Register
0x00C	LETIMERn_CNT	RWH	Counter Value Register
0x010	LETIMERn_COMP0	RWH	Compare Value Register 0
0x014	LETIMERn_COMP1	RW	Compare Value Register 1
0x018	LETIMERn_REP0	RWH	Repeat Counter Register 0
0x01C	LETIMERn_REP1	RWH	Repeat Counter Register 1
0x020	LETIMERn_IF	R	Interrupt Flag Register
0x024	LETIMERn_IFS	W1	Interrupt Flag Set Register
0x028	LETIMERn_IFC	(R)W1	Interrupt Flag Clear Register
0x02C	LETIMERn_IEN	RW	Interrupt Enable Register
0x034	LETIMERn_SYNCBUSY	R	Synchronization Busy Register
0x040	LETIMERn_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x044	LETIMERn_ROUTELOC0	RW	I/O Routing Location Register
0x050	LETIMERn_PRSSEL	RW	PRS Input Select Register

23.5 Register Description

23.5.1 LETIMERn_CTRL - Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																				0			0	0	0	0	0x0		0x0		0x0	
Access																				RW			RW	RW	RW	RW	RW		RW		RW	
Name																				DEBUGRUN			COMP0TOP	BUFTOP	OPOL1	OPOL0	UFOA1		UFOA0		REPMODE	

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	DEBUGRUN	0	RW	**Debug Mode Run Enable**

Set to keep the LETIMER running in debug mode.

Value	Description
0	LETIMER is frozen in debug mode
1	LETIMER is running in debug mode

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	COMP0TOP	0	RW	**Compare Value 0 is Top Value**

When set, the counter is cleared in the clock cycle after a compare match with compare channel 0.

Value	Description
0	The top value of the LETIMER is 65535 (0xFFFF)
1	The top value of the LETIMER is given by COMP0

Bit	Name	Reset	Access	Description
8	BUFTOP	0	RW	**Buffered Top**

Set to load COMP1 into COMP0 when REP0 reaches 0, allowing a buffered top value.

Value	Description
0	COMP0 is only written by software
1	COMP0 is set to COMP1 when REP0 reaches 0

Bit	Name	Reset	Access	Description
7	OPOL1	0	RW	**Output 1 Polarity**

Defines the idle value of output 1.

Bit	Name	Reset	Access	Description
6	OPOL0	0	RW	**Output 0 Polarity**

Defines the idle value of output 0.

Bit	Name	Reset	Access	Description
5:4	UFOA1	0x0	RW	**Underflow Output Action 1**

Defines the action on LETn_O1 on a LETIMER underflow.

Value	Mode	Description
0	NONE	LETn_O1 is held at its idle value as defined by OPOL1
1	TOGGLE	LETn_O1 is toggled on CNT underflow
2	PULSE	LETn_O1 is held active for one $LFACLK_{LETIMER0}$ clock cycle on CNT underflow. The output then returns to its idle value as defined by OPOL1
3	PWM	LETn_O1 is set idle on CNT underflow, and active on compare match with COMP1

Bit	Name	Reset	Access	Description
3:2	UFOA0	0x0	RW	**Underflow Output Action 0**

Defines the action on LETn_O0 on a LETIMER underflow.

Value	Mode	Description
0	NONE	LETn_O0 is held at its idle value as defined by OPOL0
1	TOGGLE	LETn_O0 is toggled on CNT underflow
2	PULSE	LETn_O0 is held active for one $LFACLK_{LETIMER0}$ clock cycle on CNT underflow. The output then returns to its idle value as defined by OPOL0
3	PWM	LETn_O0 is set idle on CNT underflow, and active on compare match with COMP1

Bit	Name	Reset	Access	Description
1:0	REPMODE	0x0	RW	**Repeat Mode**

Allows the repeat counter to be enabled and disabled.

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
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

23.5.2 LETIMERn_CMD - Command Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																												0	0	0	0	0
Access																												W1	W1	W1	W1	W1
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	0	W1	**Clear Toggle Output 1**
	Set to drive toggle output 1 to its idle value			
3	CTO0	0	W1	**Clear Toggle Output 0**
	Set to drive toggle output 0 to its idle value			
2	CLEAR	0	W1	**Clear LETIMER**
	Set to clear LETIMER			
1	STOP	0	W1	**Stop LETIMER**
	Set to stop LETIMER			
0	START	0	W1	**Start LETIMER**
	Set to start LETIMER			

23.5.3 LETIMERn_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																																0
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	0	R	**LETIMER Running**
	Set when LETIMER is running.			

23.5.4 LETIMERn_CNT - Counter Value 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																	0x0000															
Access																	RWH															
Name																	CNT															

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	CNT	0x0000	RWH	**Counter Value**
	Use to read the current value of the LETIMER.			

23.5.5 LETIMERn_COMP0 - Compare Value Register 0 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																	0x0000															
Access																	RWH															
Name																	COMP0															

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	COMP0	0x0000	RWH	**Compare Value 0**
	Compare and optionally top value for LETIMER.			

23.5.6 LETIMERn_COMP1 - Compare Value Register 1 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																	0x0000															
Access																	RW															
Name																	COMP1															

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	COMP1	0x0000	RW	**Compare Value 1**
	Compare and optionally buffered top value for LETIMER.			

23.5.7 LETIMERn_REP0 - Repeat Counter Register 0 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									0x00							
Access																									RWH							
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	0x00	RWH	**Repeat Counter 0**
	Optional repeat counter.			

23.5.8 LETIMERn_REP1 - Repeat Counter Register 1 (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																												0x00				
Access																												RWH				
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	0x00	RWH	**Repeat Counter 1**
	Optional repeat counter or buffer for REP0.			

23.5.9 LETIMERn_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																												0	0	0	0	0
Access																												R	R	R	R	R
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	0	R	**Repeat Counter 1 Interrupt Flag**
	Set when repeat counter 1 reaches zero.			
3	REP0	0	R	**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	0	R	**Underflow Interrupt Flag**
	Set on LETIMER underflow.			
1	COMP1	0	R	**Compare Match 1 Interrupt Flag**
	Set when LETIMER reaches the value of COMP1.			
0	COMP0	0	R	**Compare Match 0 Interrupt Flag**
	Set when LETIMER reaches the value of COMP0.			

23.5.10 LETIMERn_IFS - Interrupt Flag Set 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																												0	0	0	0	0
Access																												W1	W1	W1	W1	W1
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	0	W1	**Set REP1 Interrupt Flag**
	Write 1 to set the REP1 interrupt flag			
3	REP0	0	W1	**Set REP0 Interrupt Flag**
	Write 1 to set the REP0 interrupt flag			
2	UF	0	W1	**Set UF Interrupt Flag**
	Write 1 to set the UF interrupt flag			
1	COMP1	0	W1	**Set COMP1 Interrupt Flag**
	Write 1 to set the COMP1 interrupt flag			
0	COMP0	0	W1	**Set COMP0 Interrupt Flag**
	Write 1 to set the COMP0 interrupt flag			

23.5.11 LETIMERn_IFC - Interrupt Flag Clear 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																												0	0	0	0	0
Access																												(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
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	0	(R)W1	**Clear REP1 Interrupt Flag**
	Write 1 to clear the REP1 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	REP0	0	(R)W1	**Clear REP0 Interrupt Flag**
	Write 1 to clear the REP0 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	UF	0	(R)W1	**Clear UF Interrupt Flag**
	Write 1 to clear the UF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	COMP1	0	(R)W1	**Clear COMP1 Interrupt Flag**
	Write 1 to clear the COMP1 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	COMP0	0	(R)W1	**Clear COMP0 Interrupt Flag**
	Write 1 to clear the COMP0 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

23.5.12 LETIMERn_IEN - Interrupt Enable 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																												0	0	0	0	0
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	0	RW	**REP1 Interrupt Enable**
	Enable/disable the REP1 interrupt			
3	REP0	0	RW	**REP0 Interrupt Enable**
	Enable/disable the REP0 interrupt			
2	UF	0	RW	**UF Interrupt Enable**
	Enable/disable the UF interrupt			
1	COMP1	0	RW	**COMP1 Interrupt Enable**
	Enable/disable the COMP1 interrupt			
0	COMP0	0	RW	**COMP0 Interrupt Enable**
	Enable/disable the COMP0 interrupt			

23.5.13 LETIMERn_SYNCBUSY - Synchronization Busy 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																															0	
Access																															R	
Name																															CMD	

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	CMD	0	R	**CMD Register Busy**
	Set when the value written to CMD is being synchronized.			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

23.5.14 LETIMERn_ROUTEPEN - I/O Routing Pin 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																															0	0
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	0	RW	**Output 1 Pin Enable**

When set, output 1 of the LETIMER is enabled.

Value	Description
0	The LETn_O1 pin is disabled
1	The LETn_O1 pin is enabled

Bit	Name	Reset	Access	Description
0	OUT0PEN	0	RW	**Output 0 Pin Enable**

When set, output 0 of the LETIMER is enabled.

Value	Description
0	The LETn_O0 pin is disabled
1	The LETn_O0 pin is enabled

23.5.15 LETIMERn_ROUTELOC0 - I/O Routing Location 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																			0x00								0x00					
Access																			RW								RW					
Name																			OUT1LOC								OUT0LOC					

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*		
13:8	OUT1LOC	0x00	RW	**I/O Location**

Decides the location of the LETIMER OUT1 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	OUT0LOC	0x00	RW	**I/O Location**

Decides the location of the LETIMER OUT0 pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

23.5.16 LETIMERn_PRSSEL - PRS Input 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				0x0						0x0						0x0			
Access					RW				RW				RW				RW						RW						RW			
Name					PRSCLEARMODE				PRSSTOPMODE				PRSSTARTMODE				PRSCLEARSEL						PRSSTOPSEL						PRSSTARTSEL			

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**

Determines mode for PRS input clear.

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

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**

Determines mode for PRS input stop.

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

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**

Determines mode for PRS input start.

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
17:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:12	PRSCLEARSEL	0x0	RW	**PRS Clear Select**

Determines which PRS input can clear the LETIMER.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input
8	PRSCH8	PRS Channel 8 selected as input
9	PRSCH9	PRS Channel 9 selected as input
10	PRSCH10	PRS Channel 10 selected as input
11	PRSCH11	PRS Channel 11 selected as input
12	PRSCH12	PRS Channel 12 selected as input
13	PRSCH13	PRS Channel 13 selected as input
14	PRSCH14	PRS Channel 14 selected as input
15	PRSCH15	PRS Channel 15 selected as input

11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:6	PRSSTOPSEL	0x0	RW	**PRS Stop Select**

Determines which PRS input can stop the LETIMER.

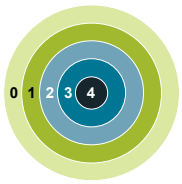
Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input

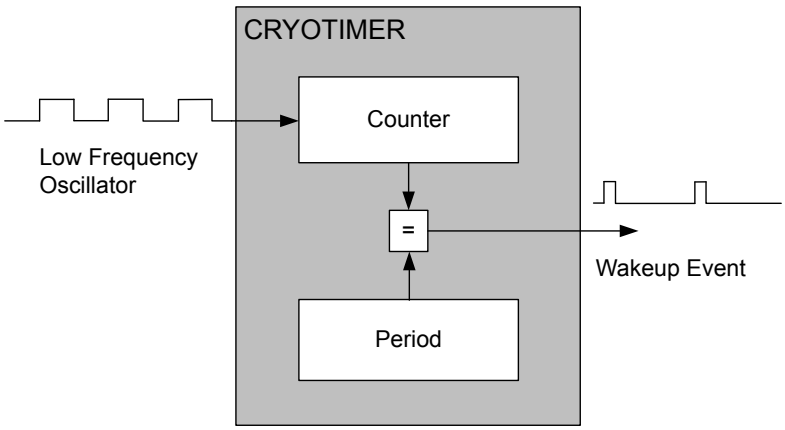
Bit	Name	Reset	Access	Description
	8	PRSCH8		PRS Channel 8 selected as input
	9	PRSCH9		PRS Channel 9 selected as input
	10	PRSCH10		PRS Channel 10 selected as input
	11	PRSCH11		PRS Channel 11 selected as input
	12	PRSCH12		PRS Channel 12 selected as input
	13	PRSCH13		PRS Channel 13 selected as input
	14	PRSCH14		PRS Channel 14 selected as input
	15	PRSCH15		PRS Channel 15 selected as input
5:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	PRSSTARTSEL	0x0	RW	**PRS Start Select**

Determines which PRS input can start the LETIMER.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input
8	PRSCH8	PRS Channel 8 selected as input
9	PRSCH9	PRS Channel 9 selected as input
10	PRSCH10	PRS Channel 10 selected as input
11	PRSCH11	PRS Channel 11 selected as input
12	PRSCH12	PRS Channel 12 selected as input
13	PRSCH13	PRS Channel 13 selected as input
14	PRSCH14	PRS Channel 14 selected as input
15	PRSCH15	PRS Channel 15 selected as input

24. CRYOTIMER - Ultra Low Energy Timer/Counter





Quick Facts

What?

The CRYOTIMER is a timer capable of providing wakeup events/interrupts after deterministic intervals in all energy modes, including EM4.

Why?

The CRYOTIMER enables the chip to remain in the lowest energy modes for long durations, while keeping track of time and being able to wake up at regular intervals, all with an absolute minimum current consumption.

How?

Using a counter running on a prescaled Low Frequency Oscillator, the CRYOTIMER can provide periodic wakeup events with a very wide period range.

24.1 Introduction

The CRYOTIMER 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 CRYOTIMER provides a very wide range of periods for the interrupts facilitating flexible ultra-low energy operation.

Because of its simplicity, the CRYOTIMER is a lower energy solution for periodically waking up the MCU compared to the RTCC.

24.2 Features

- 32 bit Counter
- Works in all the energy modes
- Only External and Power-On resets reset the CRYOTIMER
- Interrupt/wake up event after deterministic intervals
- PRS Output
- Debug mode
  - Configurable to either run or stop when processor is stopped (break)

24.3 Functional Description

24.3.1 Block Diagram

An overview of the CRYOTIMER is shown in Figure 24.1 CRYOTIMER Block Overview on page 912.

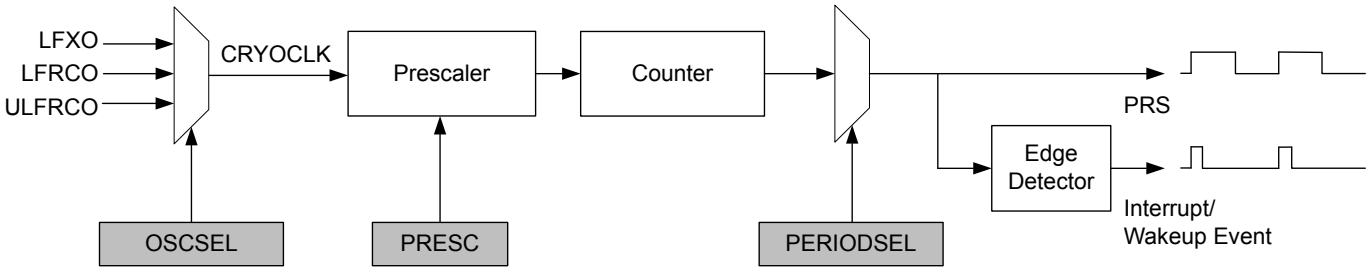


Figure 24.1. CRYOTIMER Block Overview

24.3.2 Operation

The desired low frequency oscillator for the CRYOTIMER operation can be selected by using OSCSEL in CRYOTIMER_CTRL. The selection must be made before enabling the CRYOTIMER, and it must be ensured that the selected oscillator is ready. This can be checked by observing LFXORDY or LFRCORDY (depending upon the oscillator selection) in CMU_STATUS. Note that the ULFRCO is always ready.

By default the CRYOTIMER is held in reset. It can be started by setting EN in CRYOTIMER_CTRL. The CRYOTIMER, when running, is reset by clearing EN.

The timer counts at a frequency determined by PRESC in CRYOTIMER_CTRL. This value should be set before the CRYOTIMER is enabled. Setting PRESC to 0 gives the maximum resolution, while higher values allow longer periods, see Table 24.1 CRYOTIMER Resolution vs Maximum Wakeup Event/Interrupt Period, $F_{CRYOCLK}$ = 32768 Hz on page 913.

The 32-bit Counter provides 32 different options for selecting the duration between the Wakeup events. The selected duration is specified by CRYOTIMER_PERIODSEL. It should be configured before the CRYOTIMER is enabled.

$$T_{WU} = (2^{PRESC} \times 2^{PERIODSEL})/f_{CRYOCLK}$$

Figure 24.2. Duration Between the CRYOTIMER Wakeup Events in Seconds

Table 24.1. CRYOTIMER Resolution vs Maximum Wakeup Event/Interrupt Period, $F_{CRYOCLK}$ = 32768 Hz

CRYOTIMER_CTRL_PRESC	Resolution, $2^{PRESC}/f_{CRYOCLK}$	Maximum Wakeup event/Interrupt Period
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

The 32-bit counter value of the CRYOTIMER can be read using the CRYOTIMER_CNT register.

The PRS output pulses of the CRYOTIMER are 1 CRYOCLK clock cycle wide. However, if the PRESC and PERIODSEL are both set to 0, the width of these pulses will be half CRYOCLK time period.

The CRYOTIMER wakeup events set the flag in the CRYOTIMER_IF. Interrupt on this event can be enabled by using the CRYOTIMER_IEN register.

The CRYOTIMER is always reset by the External Pin and Power-On resets. Additionally, by using EMU_CTRL, it can also be configured to reset by Watchdog, lockup, and system request resets.

Note: The CRYOTIMER configuration bits/registers should only be changed when EN in CRYOTIMER_CTRL is cleared.

24.3.3 Debug Mode

When the CPU is halted in debug mode, the CRYOTIMER can be configured to either continue to run or to be frozen. This is configured using DEBUGRUN in CRYOTIMER_CTRL.

24.3.4 Energy Mode Availability

The CRYOTIMER is available in all energy modes. Wakeup from EM2 DeepSleep and EM3 Stop to EM0 Active can be performed using the regular interrupt as discussed in 24.3.2 Operation. To generate wakeup events during EM4 Hibernate/Shutoff, EM4WU in CRYOTIMER_EM4WUEN must be set to 1. Since the interrupt flag serves as the wakeup source, it must be cleared by software after exiting a low energy mode. Refer to 9. EMU - Energy Management Unit for details on how to configure the EMU.

24.4 Register Map

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

Offset	Name	Type	Description
0x000	CRYOTIMER_CTRL	RW	Control Register
0x004	CRYOTIMER_PERIODSEL	RW	Interrupt Duration
0x008	CRYOTIMER_CNT	R	Counter Value
0x00C	CRYOTIMER_EM4WUEN	RW	Wake Up Enable
0x010	CRYOTIMER_IF	R	Interrupt Flag Register
0x014	CRYOTIMER_IFS	W1	Interrupt Flag Set Register
0x018	CRYOTIMER_IFC	(R)W1	Interrupt Flag Clear Register
0x01C	CRYOTIMER_IEN	RW	Interrupt Enable Register

24.5 Register Description

24.5.1 CRYOTIMER_CTRL - 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				0x0		0	0
Access																									RW				RW		RW	RW
Name																									PRESC				OSCSEL		DEBUGRUN	EN

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	PRESC	0x0	RW	**Prescaler Setting**

These bits select the prescaling factor.

Value	Mode	Description
0	DIV1	LF Oscillator frequency undivided
1	DIV2	LF Oscillator frequency divided by 2
2	DIV4	LF Oscillator frequency divided by 4
3	DIV8	LF Oscillator frequency divided by 8
4	DIV16	LF Oscillator frequency divided by 16
5	DIV32	LF Oscillator frequency divided by 32
6	DIV64	LF Oscillator frequency divided by 64
7	DIV128	LF Oscillator frequency divided by 128

4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:2	OSCSEL	0x0	RW	**Select Low Frequency Oscillator**

These bits select the low frequency oscillator for the CRYOTIMER operation. This field should be set after the oscillator to be selected is ready.

Value	Mode	Description
0	DISABLED	Output is driven low
1	LFRCO	Select Low Frequency RC Oscillator
2	LFXO	Select Low Frequency Crystal Oscillator
3	ULFRCO	Select Ultra Low Frequency RC Oscillator

1	DEBUGRUN	0	RW	**Debug Mode Run Enable**

Set this bit to enable CRYOTIMER to run in debug mode.

Bit	Name	Reset	Access	Description
0	EN	0	RW	**Enable CRYOTIMER**
	Set this bit to start the CRYOTIMER. Clear this bit to reset the CRYOTIMER. This bit should be set after the oscillator to be selected is ready.			

24.5.2 CRYOTIMER_PERIODSEL - Interrupt Duration

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																											0x20					
Access																											RW					
Name																											PERIODSEL					

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	PERIODSEL	0x20	RW	**Interrupts/Wakeup Events Period Setting**

Defines the duration between the Interrupts/Wakeup events based on the pre-scaled clock.

Value	Description
0	Wakeup event after every Pre-scaled clock cycle.
1	Wakeup event after 2 Pre-scaled clock cycles.
2	Wakeup event after 4 Pre-scaled clock cycles.
3	Wakeup event after 8 Pre-scaled clock cycles.
4	Wakeup event after 16 Pre-scaled clock cycles.
5	Wakeup event after 32 Pre-scaled clock cycles.
6	Wakeup event after 64 Pre-scaled clock cycles.
7	Wakeup event after 128 Pre-scaled clock cycles.
8	Wakeup event after 256 Pre-scaled clock cycles.
9	Wakeup event after 512 Pre-scaled clock cycles.
10	Wakeup event after 1k Pre-scaled clock cycles.
11	Wakeup event after 2k Pre-scaled clock cycles.
12	Wakeup event after 4k Pre-scaled clock cycles.
13	Wakeup event after 8k Pre-scaled clock cycles.
14	Wakeup event after 16k Pre-scaled clock cycles.
15	Wakeup event after 32k Pre-scaled clock cycles.
16	Wakeup event after 64k Pre-scaled clock cycles.
17	Wakeup event after 128k Pre-scaled clock cycles.
18	Wakeup event after 256k Pre-scaled clock cycles.
19	Wakeup event after 512k Pre-scaled clock cycles.
20	Wakeup event after 1M Pre-scaled clock cycles.
21	Wakeup event after 2M Pre-scaled clock cycles.
22	Wakeup event after 4M Pre-scaled clock cycles.

Bit	Name	Reset	Access	Description
	23			Wakeup event after 8M Pre-scaled clock cycles.
	24			Wakeup event after 16M Pre-scaled clock cycles.
	25			Wakeup event after 32M Pre-scaled clock cycles.
	26			Wakeup event after 64M Pre-scaled clock cycles.
	27			Wakeup event after 128M Pre-scaled clock cycles.
	28			Wakeup event after 256M Pre-scaled clock cycles.
	29			Wakeup event after 512M Pre-scaled clock cycles.
	30			Wakeup event after 1024M Pre-scaled clock cycles.
	31			Wakeup event after 2048M Pre-scaled clock cycles.
	32			Wakeup event after 4096M Pre-scaled clock cycles.

24.5.3 CRYOTIMER_CNT - Counter Value

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	0x00000000																															
Access	R																															
Name	CNT																															

Bit	Name	Reset	Access	Description
31:0	CNT	0x00000000	R	**Counter Value**
	These bits hold the Counter value.			

24.5.4 CRYOTIMER_EM4WUEN - Wake Up Enable

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																																0
Access																																RW
Name																																EM4WU

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	EM4WU	0	RW	**EM4 Wake-up Enable**
	Write 1 to enable wake-up request, write 0 to disable wake-up request.			

24.5.5 CRYOTIMER_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																																0
Access																																R
Name																																PERIOD

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	PERIOD	0	R	**Wakeup Event/Interrupt**
	Set when the Wakeup event/Interrupt occurs.			

24.5.6 CRYOTIMER_IFS - Interrupt Flag Set 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																																0
Access																																W1
Name																																PERIOD

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	PERIOD	0	W1	**Set PERIOD Interrupt Flag**
	Write 1 to set the PERIOD interrupt flag			

24.5.7 CRYOTIMER_IFC - Interrupt Flag Clear 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																																0
Access																																(R)W1
Name																																PERIOD

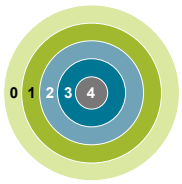
Bit	Name	Reset	Access	Description
31:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	PERIOD	0	(R)W1	**Clear PERIOD Interrupt Flag**
	Write 1 to clear the PERIOD interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

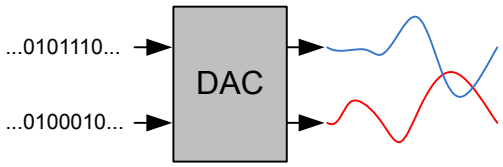
24.5.8 CRYOTIMER_IEN - Interrupt Enable Register

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																																0
Access																																RW
Name																																PERIOD

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	PERIOD	0	RW	**PERIOD Interrupt Enable**
	Enable/disable the PERIOD interrupt			

25. VDAC - Digital to Analog Converter





Quick Facts

What?

The VDAC is designed for low energy consumption, but can also provide very good performance. It can convert digital values to analog signals at up to 500 kilo samples/second with 12-bit accuracy.

Why?

The VDAC can be used to generate accurate analog signals for sound, sensors and other applications, using only a limited amount of energy.

How?

The VDAC can generate high-resolution analog signals while the MCU is operating at low frequencies and with low total power consumption. Using DMA and a timer, the VDAC can be used to generate waveforms without any CPU intervention. The VDAC is available down to Energy Mode 3.

25.1 Introduction

The Voltage Digital to Analog Converter (VDAC) can convert a digital value to an analog output voltage. The VDAC is fully differential rail-to-rail, with 12-bit resolution. It has two single ended output buffers which can be combined into one differential output. The VDAC may be used for a number of different applications such as sensor interfaces or sound output.

25.2 Features

- 500 ksamples/s operation
- Two single ended output channels
  - Can be combined into one differential output
- Integrated prescaler with division factor selectable between 1-128
- Selectable voltage reference
  - Internal low noise 2.5 V
  - Internal low noise 1.25 V
  - Internal low power 2.5 V
  - Internal low power 1.25 V
  - AVDD
  - External Pin Reference
- Conversion triggers
  - Data write
  - PRS input
  - Refresh timer
  - LESENSE
- Automatic refresh timer
  - Selection from 16-64 DAC_CLK cycles
  - Individual refresh enable for each channel
- Interrupt generation on buffer empty or finished conversion
  - Separate interrupt flags for each channel
- PRS output pulse on finished conversion
  - Separate line for each channel
- DMA request on buffer empty
  - Separate request for each channel
- Support for offset and gain calibration
- Output to dedicated pins or APORT bus
- Internal connections to ADC and ACMP
- Sine generation mode
- Asynchronous clocking mode

25.3 Functional Description

An overview of the VDAC module is shown in the figure below.

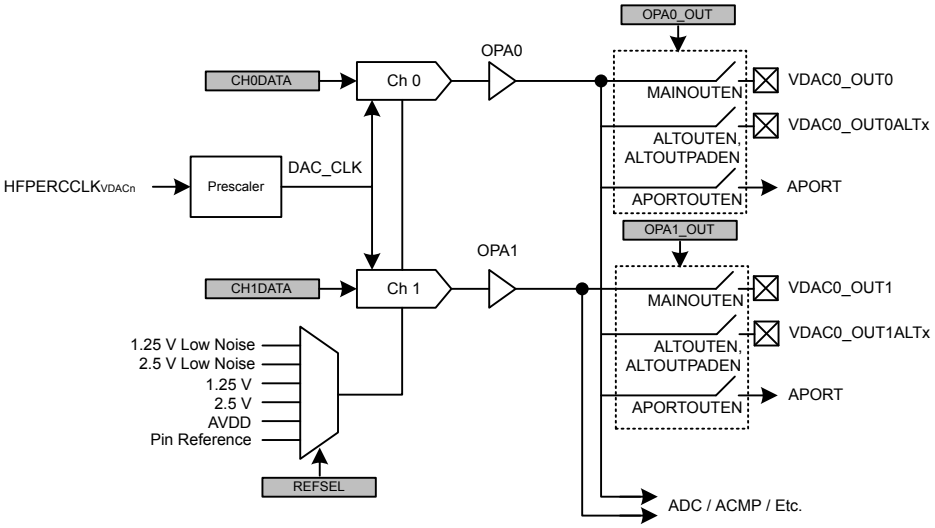


Figure 25.1. VDAC Overview

25.3.1 Power Supply

The VDAC module power (V_{OPA}) is derived from the AVDD supply pin.

25.3.2 I/O Pin Considerations

The maximum usable analog signal that can be seen on external VDAC outputs depends on several factors: whether the signal is routed through the APORT, whether overvoltage is enabled, and on the IOVDD/AVDD supply voltages, as shown in the Table 25.1 Maximum Usable IO Voltage on page 923 table.

Table 25.1. Maximum Usable IO Voltage

VDAC Pin	Maximum IO Voltage (APORT USED and OVT Enabled/ Disabled)	Maximum IO Voltage (APORT UNUSED, OVT Enabled)	Maximum IO Voltage (APORT UNUSED, OVT Disabled)
VDAC External VREF Inputs	N/A	MIN(AVDD, IOVDD)	MIN(AVDD, IOVDD)
VDAC Outputs	MIN(AVDD, IOVDD)	MIN(AVDD, IOVDD + 2 V)	MIN(AVDD, IOVDD)

25.3.3 Enabling and Disabling a Channel

A VDAC channel is enabled by writing 1 to the CHxEN and disabled by writing 1 to CHxDIS in VDACn_CMD. The channel enabled status can be read by polling the CHxENS bit in VDACn_STATUS. This bit will go high immediately following a write to CHxEN. After disabling a channel the CHxENS bit will stay high until the VDAC channel is completely disabled.

Software should configure the VDAC before enabling a channel. Software *must not* write to any of the following registers while *either* CH0ENS or CH1ENS are set:

- VDACn_CTRL
- VDACn_CHxCTRL
- VDACn_OPAxTIMER

A VDAC channel will not begin driving its output before it is enabled *and* has received a conversion trigger, see 25.3.4.3 Conversion Trigger. After a channel is enabled it will listen for trigger sources specified in TRIGMODE in VDACn_CHxCTRL. If TRIGMODE is set to SW, SWPRS or SWREFRESH and a value was written to CHxDATA or COMBDATA before enabling the channel a conversion will start immediately when the channel is enabled. When disabling a channel any pending triggers are flushed.

25.3.4 Conversions

The VDAC consists of two channels (channel 0 and 1) with separate 12-bit data registers (VDACn_CH0DATA and VDACn_CH1DATA). These can be used to produce two independent single ended outputs or the channel 0 register can be used to drive both outputs in differential mode. The VDAC supports two conversion modes: continuous and sample/off.

25.3.4.1 Continuous Mode

In continuous mode the VDAC channels will drive their outputs continuously with the data in the VDACn_CHxDATA registers. A channel is configured in continuous mode by programming the CONVMODE bitfield in VDACn_CHxCTRL to CONTINUOUS. This mode will maintain the output voltage and no manual refresh is needed.

In continuous mode the SETTLETIME field in VDACn_OPAxTIMER should be programmed to zero to achive the maximum update rate.

25.3.4.2 Sample/Off Mode

In sample/off mode the VDAC will only drive the output for a limited time per conversion. A channel is configured in sample/off mode by programming the CONVMODE bitfield in VDACn_CHxCTRL to SAMPLEOFF. How long the channel should drive the output can be controlled by programming the SETTLETIME field in the VDACn_OPAxTIMER register. The VDAC will drive the output for SETTLETIME f_{DAC_CLK} cycles before tristating the output again (and therefore if SETTLETIME is set to zero, the output will never be driven when using sample/off mode).

25.3.4.3 Conversion Trigger

Conversions can only be done while a channel is enabled, see 25.3.3 Enabling and Disabling a Channel.

If TRIGMODE is programmed to SW, SWPRS or SWREFRESH a conversion can be started by writing to the VDACn_CHxDATA register. The data registers are also mapped to a combined data register, VDACn_COMBDATA, where the data values for both channels can be written simultaneously. Writing to this register will trigger all enabled channels.

If TRIGMODE is programmed to PRS or SWPRS, a conversion can be started by an incoming pulse on the PRS channel selected in PRSSEL in VDACn_CHxCTRL. The PRSASYNC bit in VDACn_CHxCTRL determines if the VDAC expects a PRS pulse coming from a synchronous or asynchronous PRS producer.

If TRIGMODE is programmed to REFRESH or SWREFRESH a conversion will start on an overflow of the internal refresh timer. See 25.3.10 Refresh Timer.

If TRIGMODE is programmed to LESENSE a conversion will start when the LESENSE block sends a request. This setting needs to be selected whenever the channel is under LESENE control.

25.3.4.4 PRS Triggers

PRS triggers can be used to set a constant sample frequency, for instance by using a TIMER. In order to get a jitter-free sample rate, set DACCLKMODE to SYNC, set the CH0PRESCRST bit and clear the PRSASYNC bit. Note that this is only possible for channel 0.

The PRSASYNC bit tells whether the VDAC expects a synchronous PRS producer or not. When this bit is cleared, the PRS pulse must come from a synchronous producer and HFPERCLK must be running (this clock is turned off in EM2 and below). When PRSASYNC is set, the corresponding PRS channels should also bet configued as asynchronous (see the PRS chapter).

When either DACCLKMODE is set to ASYNC or the PRSASYNC bit is set, the sample frequency cannot be guaranteed to be jitter-free with respect to the PRS pulses.

The PRS frequency should never be higher than 0.5 MHz (the fastest possible sample rate). In addition the PRS frequency should not be higher than $f_{HFPERCLK}/12$ (in synchronous mode). If the PRS frequency is set too high, some PRS pulses will be dropped and the output can jitter.

25.3.5 Reference Selection

These voltage references are available and are selected by programming the REFSEL field in VDACn_CTRL.

- Internal 1.25 V Low Noise Bandgap Reference
- Internal 2.5 V Low Noise Bandgap Reference
- Internal 1.25 V Low Power Bandgap Reference
- Internal 2.5 V Low Power Bandgap Reference
- AVDD

- External Pin

25.3.6 Warmup Time and Initial Conversion

When a channel is first enabled it needs to warm up. This is performed automatically during the first conversion. The time required to warm up depends on the programmed DRIVESTRENGTH field in VDACn_OPAx_CTRL. In Table 25.2 VDAC Warmup Time on page 925 the minimum WARMUPTIME field for each drive strength is specified. Software is responsible for programming the correct value to WARMUPTIME before enabling a channel. If the time is programmed too short, an undefined voltage may be output until the VDAC settles.

The CHxWARM bits in VDACn_STATUS are set when the warmup period has completed.

A consequence of the warmup period is that in continuous mode, the first conversion might take longer than the following conversions. In order to make sure all samples have the same timing, perform a dummy conversion to make the VDAC settle to a known voltage first.

Table 25.2. VDAC Warmup Time

DRIVESTRENGTH	WARMUPTIME
0	100 µs
1	85 µs
2	8 µs
3	8 µs

25.3.7 Analog Output

The output selection for each VDAC channel is configured in the VDACn_OPAx_OUT registers. Each VDAC channel has its own main output pin, VDACn_OUTx, that can be enabled with MAINOUTEN. In addition, several alternate outputs can be selected. These are enabled by first setting ALTOUTEN and then setting the corresponding bit(s) in ALTOUTPADEN. The VDAC output can also be routed to APORT by setting APORTOUTEN and configuring the APORTOUTSEL field to select the desired APORT.

The VDAC outputs also have direct internal connections to ADCs and ACMPs. These outputs are always enabled and can be selected by configuring the input selection for the ADC/ACMP.

In sample/off mode the VDAC will only drive the output for the duration programmed in SETTLETIME (in VDACn_OPAx_TIMER register) for each incoming conversion trigger. In continuous mode the VDAC will continue to drive the output until the channel is disabled. However, note that also in this mode a conversion trigger is needed before the output is enabled. See 25.3.3 Enabling and Disabling a Channel and 25.3.4.3 Conversion Trigger.

25.3.8 Output Mode

The two VDAC channels can act as two separate single ended channels or be combined into one differential channel. This is selected through the DIFF bit in VDACn_CTRL.

25.3.8.1 Single Ended Output

When operating in single ended mode, the channel 0 output is on VDACn_OUT0 and the channel 1 output is on VDACn_OUT1. The output voltage can be calculated using Figure 25.2 VDAC Single Ended Output Voltage on page 925

$$V_{OUT} = V_{VDACn\_OUTx} - V_{SS} = V_{ref} \times CHxDATA/4095$$

Figure 25.2. VDAC Single Ended Output Voltage

where CHxDATA is a 12-bit unsigned integer.

25.3.8.2 Differential Output

When operating in differential mode, both VDAC outputs are used. The differential conversion uses VDACn_CH0DATA as source. The positive output is on VDACn_OUT1 and the negative output is on VDACn_OUT0. Since the output can be negative, it is expected that

the data is written in 2's complement form with the MSB of the 12-bit value being the signed bit. The output voltage can be calculated using Figure 25.3 VDAC Differential Output Voltage on page 926:

$$V_{OUT} = V_{VDACn\_OUT1} - V_{VDACn\_OUT0} = V_{ref} \times CH0DATA/2047$$

Figure 25.3. VDAC Differential Output Voltage

where CH0DATA is a 12-bit signed integer. The common mode voltage is $V_{ref}/2$.

When using differential mode, the user must make sure that both channels are set up identically. I.e. VDACn_CH0CTRL and VDACn_CH1CTRL must be programmed to identical values (with the exception that the PRSSEL bitfield is allowed to be programmed differently for usage together with the OUTENPRS feature). Similarly the user must program VDACn_OPA0TIMER and VDACn_OPA1TIMER to identical values.

25.3.9 Async Mode

The VDAC is default clocked from HFPERCCLK, which is automatically turned off in EM2/3. In order to allow VDAC operation in EM2/3 an internal oscillator can be selected for the VDAC by setting the DACCLKMODE bitfield in VDACn_CTRL to ASYNC. Before entering EM2/3 software must make sure the channel is enabled first by polling CHxENS in VDACn_STATUS. Entering EM2/3 with an enabled VDAC channel while DACCLKMODE is set to SYNC is a programming error and will lead to EM23ERRIF getting set to 1.

In asynchronous mode both VDAC channels are not necessarily triggered synchronous to each other and therefore the user should not assume that e.g. PRS, refresh or VDACn_COMBDATA based conversion triggers are observed by both channels at the same time. In differential mode both channels will operate in lock step, even while using the asynchronous clocking mode.

25.3.10 Refresh Timer

The VDAC incluces an internal refresh timer. The refresh timer is automatically started if a channel selects either REFRESH or SWREFRESH for TRIGMODE and the channel is enabled. The refresh timer will count the number of f_{DAC_CLK} cycles programmed in REFRESHPERIOD before wrapping and generating a conversion trigger.

25.3.11 Clock Prescaling

The VDAC has an internal clock prescaler, which can divide the input clock by any factor between 1 and 128, by setting the PRESC field in VDACn_CTRL. The resulting DAC_CLK is used by the converter core and the frequency is given by Figure 25.4 VDAC Clock Prescaling on page 926 :

$$f_{DAC\_CLK} = f_{IN\_CLK} / (PRESC + 1)$$

Figure 25.4. VDAC Clock Prescaling

where f_{IN_CLK} is the input clock frequency. The f_{DAC_CLK} must be programmed to be at most 1 MHz. When the DACCLKMODE is set to SYNC, the input clock frequency is $f_{HFPERCCLK}$. When DACCLKMODE is set to ASYNC, an internal 12Mhz oscillator is used. In this mode it is required that the PRESC field be program to 11 or higher.

The prescaler runs continuously when either of the channels are enabled. When running with a prescaler setting higher than 0, there will be an unpredictable delay from the time the conversion was triggered to the time the actual conversion takes place. This is because the conversions are controlled by the prescaled clock and the conversion can arrive at any time during a prescaled clock (DAC_CLK) period. A second reason for unpredictable delay between a trigger and the associated conversion is that the activity on one channel can impact whether the VDAC reference is warm or not and therefore it can impact whether warmup is required when using the other channel. The uncertainty related to the clock prescaler can be addressed by using CH0PRESCRST. If the CH0PRESCRST bit in VDACn_CTRL is set, the prescaler will be reset every time a conversion is triggered on channel 0. This leads to a predictable latency between channel 0 trigger and conversion (assuming the warmup sequence is deterministic as well). If channel 0 is used in continuous mode, the warmup sequence will only apply to its first conversion and software can use the CH0WARM status bit to determine if the VDAC has warmed up.

25.3.12 High Speed

The VDAC is able to do conversions up to 400 ksamples/s. In order to reach the maximum conversion rate it is recommended to configure the VDAC in the following way:

1. Make f_{DAC_CLK} 1 Mhz
2. Set TRIGMODE to SW
3. Program SETTLETIME in OPAx_TIMER to 0

4. Set up a DMA transfer from a buffer in RAM to CHxDATA
5. Set CONVMODE to CONTINUOUS

25.3.13 Sine Generation Mode

The VDAC contains an automatic sine-generation mode, which is enabled by setting the SINEMODE bit in VDACn_CTRL. In this mode, the VDAC data is overridden with a conversion data taken from a sine lookup table. The sine signal is controlled by the PRS line selected by CH0PRSSEL in VDACn_CH0CTRL. When the line is high, a sine wave will be produced. Each period, starting at 0 degrees, is made up of 16 samples and the frequency is given by Figure 25.5 VDAC Sine Generation on page 927. In case OUTENPRS equals 1, lowering the PRS line selected by CH0PRSSEL will reset the sine output to 0 degrees resulting in a voltage of Vref/2 on the output channel. In case OUTENPRS equals 0, lowering the PRS line selected by CH0PRSSEL will stop progress of the sine wave at the sample currently being output (and the sine will therefore not be reset to 0 degrees when raising the PRS line again).

$$f_{sine} = f_{HFPERCCLK} / 32 \times (PRESC + 1)$$

Figure 25.5. VDAC Sine Generation

Sine mode is supported only for the fastest configuration of the VDAC in continuous mode. Therefore the CONVMODE bitfield needs to be set to CONTINUOUS and the SETTLETIME bitfield in VDACn_OPAxTIMER need to be programmed to zero for the used channel(s) in order to use sine generation mode. The TRIGMODE bitfield needs to be programmed to PRS for any channel used for sine generation mode. The other trigger modes are not supported.

The SINE wave will be output on channel 0 and therefore requires that this channel is enabled by writing 1 to CH0EN in the VDACn_CMD register. If DIFF is set in VDACn_CTRL, the sine wave will be output on both channels, but inverted. Note that when OUTENPRS in VDACn_CTRL is set, the sine output will be reset to 0 degrees when the PRS line selected by CH1PRSSEL is low.

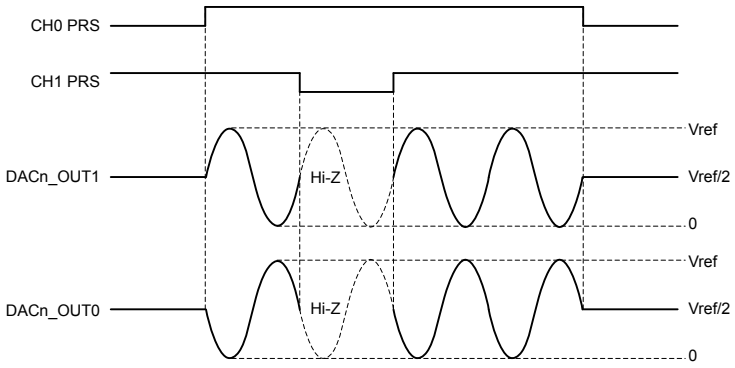


Figure 25.6. VDAC Sine Mode

25.3.14 Interrupt Flags

The VDAC has several interrupt flags, indicating state transitions and error conditions.

In addition to the VDAC interrupt flags the VDAC registers contain interrupt flags for the OPAMP modules. See The OPAMP chapter for more information on these flags.

25.3.14.1 Conversion Done

The Conversion Done (CHxCD) interrupt flags are set when a conversion is complete. The flags are set after a channel has driven the output with the new code for the time programmed in SETTLETIME in VDACn_OPAxTIMER.

25.3.14.2 Buffer Level

The Buffer Level (CHxBL) interrupt flags are set when there is space available in CHxDATA. These flags are initially set, get cleared when CHxDATA is written and set again when the value is used for a conversion.

25.3.14.3 Overflow/Underflow

If CHxDATA is written to while CHxBL is cleared, the channel overflow flag (CHxOF) will be set. If a new conversion is triggered (e.g. via PRS) before data is written to CHxDATA (CHxDATA is empty) the channel underflow flag (CHxUF) will be set.

25.3.14.4 EM2/3 Sleep Error

The VDAC can only operate in EM2/3 when DACCLKMODE is set to ASYNC. If EM2 or EM3 is entered while a channel is enabled and DACCLKMODE is set to SYNC the EM23ERRIF flag will be set.

25.3.15 PRS Outputs

The VDAC has two PRS outputs which will carry a one cycle (HFPERCCLK) high pulse when the corresponding channel has finished a conversion. Only available when DACCLKMODE is set to SYNC.

25.3.16 DMA Request

Each channel sends a DMA request when there is space in the channel's data register (VDACn_CHxDATA). These registers are initially empty and also become empty every time a conversion is triggered. The request is cleared when VDACn_CHxDATA is written.

25.3.17 LESENSE Trigger Mode

The VDAC can be controlled by LESENSE by programming the TRIGMODE field in VDACn_CHxCTRL to LESENSE. In LESENSE mode the conversion data can come from either VDACn_CHxDATA registers or LESENSE registers, depending on the LESENSE configuration. The trigger events are also controlled by the LESENSE state machine. See the LESENSE chapter for more information.

25.3.18 Opamps

The VDAC includes a set of highly configurable opamps that can be accessed with the VDAC registers. OPA0 and OPA1 is used for the output stages of the two VDAC channels, but can be used as standalone opamps if the VDAC channels are not in use. Opamps with higher numbers are completely standalone. For a detailed description see the OPAMP chapter.

25.3.19 Calibration

The VDAC contains a calibration register, VDACn_CAL, where calibration values for both offset and gain correction can be written. The required (gain) calibration values depend on the chosen reference and on whether the main or alternative VDAC output is used. The Device Information page provides the required trim values depending on reference choice and output selection in the DEVINFO_VDACnMAINCAL, DEVINFO_VDACnALTCAL, and DEVINFO_VDACnCH1CAL locations.

The OPAMPs contain a calibration register, VDACn_OPAx_CAL, where calibration values for both offset and gain correction can be written. The required calibration settings depend on the chosen DRIVESTRENGTH. The required calibration values can be found in the Device Information pages. For a given OPAMP x, the calibration settings for DRIVESTRENGTH n can be found in DEVINFO_OPAxCALn.

25.3.19.1 Channel 1 Calibration

For channel 1, the factory calibration values are only accurate for the main output. When using the alternative outputs or APORT, the error on the output may be larger than the data sheet values (even when loading values from DEVINFO_VDACn_ALTCAL). To get accurate output from channel 1, either use the main output or perform manual calibration.

25.3.19.2 Manual Calibration

To manually calibrate the VDAC:

1. Enable CH0 and CH1 in their desired modes
2. Set both channel outputs to 80% of full-scale by setting VDACn_CHxDATA = 0xCCC
3. Measure CH0 output and sweep VDACn_CAL.GAINERRTRIM until the smallest calibration error is found
4. Measure CH1 output and sweep VDACn_CAL.GAINERRTRIMCH1 until the smallest calibration error is found

The calibration error is given by

$$e = abs(V_{out}/(V_{REF} * 0.8) - 1)$$

Figure 25.7. Calibration Error

where V_{out} is the measured voltage at the pin and V_{REF} is the reference voltage.

Note that even if only CH1 is going to be used, the full calibration procedure should be followed. It is permissible to skip CH1 calibration if only CH0 is used. The following parameters influence the calibration. A change in any of these might require a re-calibration:

- VDACn_CTRL.REFSEL
- VDACn_OPAx_OUT.MAINOUTEN
- VDACn_OPAx_OUT.ALTOUTEN
- VDACn_OPAx_CTRL.DRIVESTRENGTH

25.3.20 Warmup Mode

If the WARMUPMODE field in VDACn_CTRL is set to KEEPINSTANDBY, the VDAC keeps internal bias currents running between conversions. It does not reduce the startup time, but it can help reduce noise from the VDAC to other analog peripherals, like the ADC or ACMP.

25.4 Register Map

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

Offset	Name	Type	Description
0x000	VDACn_CTRL	RW	Control Register
0x004	VDACn_STATUS	R	Status Register
0x008	VDACn_CH0CTRL	RW	Channel 0 Control Register
0x00C	VDACn_CH1CTRL	RW	Channel 1 Control Register
0x010	VDACn_CMD	W1	Command Register
0x014	VDACn_IF	R	Interrupt Flag Register
0x018	VDACn_IFS	W1	Interrupt Flag Set Register
0x01C	VDACn_IFC	(R)W1	Interrupt Flag Clear Register
0x020	VDACn_IEN	RW	Interrupt Enable Register
0x024	VDACn_CH0DATA	RWH	Channel 0 Data Register
0x028	VDACn_CH1DATA	RWH	Channel 1 Data Register
0x02C	VDACn_COMBDATA	W	Combined Data Register
0x030	VDACn_CAL	RW	Calibration Register
0x0A0	VDACn_OPA0_APORTREQ	R	Operational Amplifier APORT Request Status Register
0x0A4	VDACn_OPA0_APORTCON-FLICT	R	Operational Amplifier APORT Conflict Status Register
0x0A8	VDACn_OPA0_CTRL	RW	Operational Amplifier Control Register
0x0AC	VDACn_OPA0_TIMER	RW	Operational Amplifier Timer Control Register
0x0B0	VDACn_OPA0_MUX	RW	Operational Amplifier Mux Configuration Register
0x0B4	VDACn_OPA0_OUT	RW	Operational Amplifier Output Configuration Register
0x0B8	VDACn_OPA0_CAL	RW	Operational Amplifier Calibration Register
...	VDACn_OPAx_APORTREQ	R	Operational Amplifier APORT Request Status Register
...	VDACn_OPAx_APORTCON-FLICT	R	Operational Amplifier APORT Conflict Status Register
...	VDACn_OPAx_CTRL	RW	Operational Amplifier Control Register
...	VDACn_OPAx_TIMER	RW	Operational Amplifier Timer Control Register
...	VDACn_OPAx_MUX	RW	Operational Amplifier Mux Configuration Register
...	VDACn_OPAx_OUT	RW	Operational Amplifier Output Configuration Register
...	VDACn_OPAx_CAL	RW	Operational Amplifier Calibration Register
0x100	VDACn_OPA3_APORTREQ	R	Operational Amplifier APORT Request Status Register
0x104	VDACn_OPA3_APORTCON-FLICT	R	Operational Amplifier APORT Conflict Status Register
0x108	VDACn_OPA3_CTRL	RW	Operational Amplifier Control Register
0x10C	VDACn_OPA3_TIMER	RW	Operational Amplifier Timer Control Register
0x110	VDACn_OPA3_MUX	RW	Operational Amplifier Mux Configuration Register
0x114	VDACn_OPA3_OUT	RW	Operational Amplifier Output Configuration Register

Offset	Name	Type	Description
0x118	VDACn_OPA3_CAL	RW	Operational Amplifier Calibration Register

25.5 Register Description

25.5.1 VDACn_CTRL - 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	0			0			0x0						0x00										0x0			0	0	0				0
Access	RW			RW			RW						RW										RW			RW	RW	RW				RW
Name	DACCLKMODE			WARMUPMODE			REFRESHPERIOD						PRESC										REFSEL			CH0PRESCRST	OUTENPRS	SINEMODE				DIFF

Bit	Name	Reset	Access	Description
31	DACCLKMODE	0	RW	**Clock Mode**

Selects DAC clock source from synchronous or asynchronous - with respect to Peripheral Clock - clock source

Value	Mode	Description
0	SYNC	Uses HFPERCCLK to generate DAC_CLK, DAC will run with static settings in EM2 in this mode
1	ASYNC	Uses internal VDAC oscillator to generate DAC_CLK. DAC will be available in EM2

Bit	Name	Reset	Access	Description
30:29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28	WARMUPMODE	0	RW	**Warm-up Mode**

Select Warm-up Mode for DAC

Value	Mode	Description
0	NORMAL	DAC is shut off after each sample off conversion
1	KEEPINSTANDBY	DAC is kept in standby mode between sample off conversions

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	REFRESHPERIOD	0x0	RW	**Refresh Period**

Select refresh counter period. A channel x will be refreshed with the period set in REFRESHPERIOD if the channel in VDACn_CHxCTRL has its TRIGMODE set to REFRESH or SWREFRESH.

Value	Mode	Description
0	8CYCLES	All channels with enabled refresh are refreshed every 8 DAC_CLK cycles
1	16CYCLES	All channels with enabled refresh are refreshed every 16 DAC_CLK cycles
2	32CYCLES	All channels with enabled refresh are refreshed every 32 DAC_CLK cycles

Bit	Name	Reset	Access	Description
	3	64CYCLES		All channels with enabled refresh are refreshed every 64 DAC_CLK cycles
23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:16	PRESC	0x00	RW	**Prescaler Setting for DAC Clock**
	Selected DAC clock source (as selected by DACCLKMODE) is prescaled by PRESC+1 to generated DAC clock (DAC_CLK)			
	Value	Description		
	PRESC	Clock division factor of PRESC+1.		
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	REFSEL	0x0	RW	**Reference Selection**
	Select reference			
	Value	Mode		Description
	0	1V25LN		Internal low noise 1.25 V bandgap reference
	1	2V5LN		Internal low noise 2.5 V bandgap reference
	2	1V25		Internal 1.25 V bandgap reference
	3	2V5		Internal 2.5 V bandgap reference
	4	VDD		AVDD reference
	6	EXT		External pin reference
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	CH0PRESCRST	0	RW	**Channel 0 Start Reset Prescaler**
	Select if prescaler (determining DAC_CLK rate) is reset on channel 0 start.			
	Value			Description
	0			Prescaler not reset on channel 0 start
	1			Prescaler reset on channel 0 start
5	OUTENPRS	0	RW	**PRS Controlled Output Enable**
	Enable PRS Control of DAC output enable.			
	Value			Description
	0			DAC output enable always on
	1			DAC output enable controlled by PRS signal selected for CH1
4	SINEMODE	0	RW	**Sine Mode**
	Enable/disable sine mode.			
	Value			Description

Bit	Name	Reset	Access	Description
	0			Sine mode disabled. Sine reset to 0 degrees
	1			Sine mode enabled
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	DIFF	0	RW	**Differential Mode**
	Select single ended or differential mode.			

Value	Description
0	Single ended output
1	Differential output

25.5.2 VDACn_STATUS - Status 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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0	1	1	0	0
Access	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R											R	R	R	R	R	R
Name	OPA3OUTVALID	OPA2OUTVALID	OPA1OUTVALID	OPA0OUTVALID	OPA3WARM	OPA2WARM	OPA1WARM	OPA0WARM	OPA3ENS	OPA2ENS	OPA1ENS	OPA0ENS	OPA3APORTCONFLICT	OPA2APORTCONFLICT	OPA1APORTCONFLICT	OPA0APORTCONFLICT											CH1WARM	CH0WARM	CH1BL	CH0BL	CH1ENS	CH0ENS

Bit	Name	Reset	Access	Description
31	OPA3OUTVALID	0	R	**OPA3 Output Valid Status**
	OPA3 output is settled externally at the load. In PRS triggered mode this status flag is not used (and remains 0).			
30	OPA2OUTVALID	0	R	**OPA2 Output Valid Status**
	OPA2 output is settled externally at the load. In PRS triggered mode this status flag is not used (and remains 0).			
29	OPA1OUTVALID	0	R	**OPA1 Output Valid Status**
	OPA1 output is settled externally at the load. In PRS triggered mode this status flag is not used (and remains 0).			
28	OPA0OUTVALID	0	R	**OPA0 Output Valid Status**
	OPA0 output is settled externally at the load. In PRS triggered mode this status flag is not used (and remains 0).			
27	OPA3WARM	0	R	**OPA3 Warm Status**
	OPA3 is warm and output is enabled. In PRS triggered mode this status flag is not used (and remains 0).			
26	OPA2WARM	0	R	**OPA2 Warm Status**
	OPA2 is warm and output is enabled. In PRS triggered mode this status flag is not used (and remains 0).			
25	OPA1WARM	0	R	**OPA1 Warm Status**
	OPA1 is warm and output is enabled. In PRS triggered mode this status flag is not used (and remains 0).			
24	OPA0WARM	0	R	**OPA0 Warm Status**
	OPA0 is warm and output is enabled. In PRS triggered mode this status flag is not used (and remains 0).			
23	OPA3ENS	0	R	**OPA3 Enabled Status**
	This bit is set when OPA3 is enabled			
22	OPA2ENS	0	R	**OPA2 Enabled Status**
	This bit is set when OPA2 is enabled			
21	OPA1ENS	0	R	**OPA1 Enabled Status**
	This bit is set when OPA1 is enabled			
20	OPA0ENS	0	R	**OPA0 Enabled Status**
	This bit is set when OPA0 is enabled			

Bit	Name	Reset	Access	Description
19	OPA3APORTCON-FLICT	0	R	**OPA3 Bus Conflict Output** 1 if any of the APORTs being requested by the OPA3 are also being requested by another peripheral.
18	OPA2APORTCON-FLICT	0	R	**OPA2 Bus Conflict Output** 1 if any of the APORTs being requested by the OPA2 are also being requested by another peripheral.
17	OPA1APORTCON-FLICT	0	R	**OPA1 Bus Conflict Output** 1 if any of the APORTs being requested by the OPA1 are also being requested by another peripheral.
16	OPA0APORTCON-FLICT	0	R	**OPA0 Bus Conflict Output** 1 if any of the APORTs being requested by the OPA0 are also being requested by another peripheral.
15:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CH1WARM	0	R	**Channel 1 Warm** This bit is set when channel 1 is warm.
4	CH0WARM	0	R	**Channel 0 Warm** This bit is set when channel 0 is warm.
3	CH1BL	1	R	**Channel 1 Buffer Level** This bit is set when there is space for new data in CH1DATA.
2	CH0BL	1	R	**Channel 0 Buffer Level** This bit is set when there is space for new data in CH0DATA.
1	CH1ENS	0	R	**Channel 1 Enabled Status** This bit is set when channel 1 is enabled.
0	CH0ENS	0	R	**Channel 0 Enabled Status** This bit is set when channel 0 is enabled.

25.5.3 VDACn_CH0CTRL - Channel 0 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							0		0x0						0
Access																	RW							RW		RW						RW
Name																	PRSSEL							PRSASYNC		TRIGMODE						CONVMODE

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	PRSSEL	0x0	RW	**Channel 0 PRS Trigger Select**

Select Channel 0 PRS input channel.

Value	Mode	Description
0	PRSCH0	PRS ch 0 triggers a conversion.
1	PRSCH1	PRS ch 1 triggers a conversion.
2	PRSCH2	PRS ch 2 triggers a conversion.
3	PRSCH3	PRS ch 3 triggers a conversion.
4	PRSCH4	PRS ch 4 triggers a conversion.
5	PRSCH5	PRS ch 5 triggers a conversion.
6	PRSCH6	PRS ch 6 triggers a conversion.
7	PRSCH7	PRS ch 7 triggers a conversion.
8	PRSCH8	PRS ch 8 triggers a conversion.
9	PRSCH9	PRS ch 9 triggers a conversion.
10	PRSCH10	PRS ch 10 triggers a conversion.
11	PRSCH11	PRS ch 11 triggers a conversion.
12	PRSCH12	PRS ch 12 triggers a conversion.
13	PRSCH13	PRS ch 13 triggers a conversion.
14	PRSCH14	PRS ch 14 triggers a conversion.
15	PRSCH15	PRS ch 15 triggers a conversion.

11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	PRSASYNC	0	RW	**Channel 0 PRS Asynchronous Enable**

Set this bit to 1 to treat PRS channel as asynchronous

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

Bit	Name	Reset	Access	Description
6:4	TRIGMODE	0x0	RW	**Channel 0 Trigger Mode**
	Select Channel 0 conversion trigger.			

Value	Mode	Description
0	SW	Channel 0 is triggered by CH0DATA or COMBDATA write
1	PRS	Channel 0 is triggered by PRS input
2	REFRESH	Channel 0 is triggered by Refresh timer
3	SWPRS	Channel 0 is triggered by CH0DATA/COMBDATA write or PRS input
4	SWREFRESH	Channel 0 is triggered by CH0DATA/COMBDATA write or Refresh timer
5	LESENSE	Channel 0 is triggered by LESENSE

3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CONVMODE	0	RW	**Conversion Mode**
	Configure conversion mode.			

Value	Mode	Description
0	CONTINUOUS	DAC channel 0 is set in continuous mode
1	SAMPLEOFF	DAC channel 0 is set in sample/off mode

25.5.4 VDACn_CH1CTRL - Channel 1 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							0		0x0						0
Access																	RW							RW		RW						RW
Name																	PRSSEL							PRSASYNC		TRIGMODE						CONVMODE

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	PRSSEL	0x0	RW	**Channel 1 PRS Trigger Select**

Select Channel 1 PRS input channel.

Value	Mode	Description
0	PRSCH0	PRS ch 0 triggers a conversion.
1	PRSCH1	PRS ch 1 triggers a conversion.
2	PRSCH2	PRS ch 2 triggers a conversion.
3	PRSCH3	PRS ch 3 triggers a conversion.
4	PRSCH4	PRS ch 4 triggers a conversion.
5	PRSCH5	PRS ch 5 triggers a conversion.
6	PRSCH6	PRS ch 6 triggers a conversion.
7	PRSCH7	PRS ch 7 triggers a conversion.
8	PRSCH8	PRS ch 8 triggers a conversion.
9	PRSCH9	PRS ch 9 triggers a conversion.
10	PRSCH10	PRS ch 10 triggers a conversion.
11	PRSCH11	PRS ch 11 triggers a conversion.
12	PRSCH12	PRS ch 12 triggers a conversion.
13	PRSCH13	PRS ch 13 triggers a conversion.
14	PRSCH14	PRS ch 14 triggers a conversion.
15	PRSCH15	PRS ch 15 triggers a conversion.

11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	PRSASYNC	0	RW	**Channel 1 PRS Asynchronous Enable**

Set this bit to 1 to treat PRS channel as asynchronous

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

Bit	Name	Reset	Access	Description
6:4	TRIGMODE	0x0	RW	**Channel 1 Trigger Mode**

Select Channel 1 conversion trigger.

Value	Mode	Description
0	SW	Channel 1 is triggered by CH1DATA or COMBDATA write
1	PRS	Channel 1 is triggered by PRS input
2	REFRESH	Channel 1 is triggered by Refresh timer
3	SWPRS	Channel 1 is triggered by CH1DATA/COMBDATA write or PRS input
4	SWREFRESH	Channel 1 is triggered by CH1DATA/COMBDATA write or Refresh timer
5	LESENSE	Channel 1 is triggered by LESENSE

3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CONVMODE	0	RW	**Conversion Mode**

Configure conversion mode.

Value	Mode	Description
0	CONTINUOUS	DAC channel 1 is set in continuous mode
1	SAMPLEOFF	DAC channel 1 is set in sample/off mode

25.5.5 VDACn_CMD - 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									0	0	0	0	0	0	0	0													0	0	0	0
Access									W1	W1	W1	W1	W1	W1	W1	W1													W1	W1	W1	W1
Name									OPA3DIS	OPA3EN	OPA2DIS	OPA2EN	OPA1DIS	OPA1EN	OPA0DIS	OPA0EN													CH1DIS	CH1EN	CH0DIS	CH0EN

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	OPA3DIS	0	W1	**OPA3 Disable**
	Disables OPA3.			
22	OPA3EN	0	W1	**OPA3 Enable**
	Enables OPA3			
21	OPA2DIS	0	W1	**OPA2 Disable**
	Disables OPA2.			
20	OPA2EN	0	W1	**OPA2 Enable**
	Enables OPA2			
19	OPA1DIS	0	W1	**OPA1 Disable**
	Disables OPA1.			
18	OPA1EN	0	W1	**OPA1 Enable**
	Enables OPA1			
17	OPA0DIS	0	W1	**OPA0 Disable**
	Disables OPA0.			
16	OPA0EN	0	W1	**OPA0 Enable**
	Enables OPA0			
15:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	CH1DIS	0	W1	**DAC Channel 1 Disable**
	Disables DAC Channel 1			
2	CH1EN	0	W1	**DAC Channel 1 Enable**
	Enables DAC Channel 1.			
1	CH0DIS	0	W1	**DAC Channel 0 Disable**
	Disables DAC Channel 0.			
0	CH0EN	0	W1	**DAC Channel 0 Enable**
	Enables DAC Channel 0			

25.5.6 VDACn_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	0	0	0	0					0	0	0	0	0	0	0	0	0								1	1	0	0	0	0	0	0
Access	R	R	R	R					R	R	R	R	R	R	R	R	R								R	R	R	R	R	R	R	R
Name	OPA3OUTVALID	OPA2OUTVALID	OPA1OUTVALID	OPA0OUTVALID					OPA3PRSTIMEDERR	OPA2PRSTIMEDERR	OPA1PRSTIMEDERR	OPA0PRSTIMEDERR	OPA3APORTCONFLICT	OPA2APORTCONFLICT	OPA1APORTCONFLICT	OPA0APORTCONFLICT	EM23ERR								CH1BL	CH0BL	CH1UF	CH0UF	CH1OF	CH0OF	CH1CD	CH0CD

Bit	Name	Reset	Access	Description
31	OPA3OUTVALID	0	R	**OPA3 Output Valid Interrupt Flag**
	OPA3 output is settled externally at the load			
30	OPA2OUTVALID	0	R	**OPA3 Output Valid Interrupt Flag**
	OPA2 output is settled externally at the load			
29	OPA1OUTVALID	0	R	**OPA1 Output Valid Interrupt Flag**
	OPA1 output is settled externally at the load			
28	OPA0OUTVALID	0	R	**OPA0 Output Valid Interrupt Flag**
	OPA0 output is settled externally at the load			
27:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23	OPA3PRSTIMEDERR	0	R	**OPA3 PRS Trigger Mode Error Interrupt Flag**
	Indicates that in TIMED PRS triggered mode, the negative edge of the PRS pulse came before the OPA output was valid.			
22	OPA2PRSTIMEDERR	0	R	**OPA2 PRS Trigger Mode Error Interrupt Flag**
	Indicates that in TIMED PRS triggered mode, the negative edge of the PRS pulse came before the OPA output was valid.			
21	OPA1PRSTIMEDERR	0	R	**OPA1 PRS Trigger Mode Error Interrupt Flag**
	Indicates that in TIMED PRS triggered mode, the negative edge of the PRS pulse came before the OPA output was valid.			
20	OPA0PRSTIMEDERR	0	R	**OPA0 PRS Trigger Mode Error Interrupt Flag**
	Indicates that in TIMED PRS triggered mode, the negative edge of the PRS pulse came before the OPA output was valid.			
19	OPA3APORTCONFLICT	0	R	**OPA3 Bus Conflict Output Interrupt Flag**
	1 if any of the APORTs being requested by the OPA3 are also being requested by another peripheral.			
18	OPA2APORTCONFLICT	0	R	**OPA2 Bus Conflict Output Interrupt Flag**
	1 if any of the APORTs being requested by the OPA0 are also being requested by another peripheral.			

Bit	Name	Reset	Access	Description
17	OPA1APORTCON-FLICT	0	R	**OPA1 Bus Conflict Output Interrupt Flag**
	1 if any of the APORTs being requested by the OPA1 are also being requested by another peripheral.			
16	OPA0APORTCON-FLICT	0	R	**OPA0 Bus Conflict Output Interrupt Flag**
	1 if any of the APORTs being requested by the OPA0 are also being requested by another peripheral.			
15	EM23ERR	0	R	**EM2/3 Entry Error Flag**
	Set when going to EM2/3 while DACCLKMODE equals SYNC and a channel is enabled			
14:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	CH1BL	1	R	**Channel 1 Buffer Level Interrupt Flag**
	Indicates space available in CH1DATA.			
6	CH0BL	1	R	**Channel 0 Buffer Level Interrupt Flag**
	Indicates space available in CH0DATA.			
5	CH1UF	0	R	**Channel 1 Data Underflow Interrupt Flag**
	Indicates channel 1 data underflow.			
4	CH0UF	0	R	**Channel 0 Data Underflow Interrupt Flag**
	Indicates channel 0 data underflow.			
3	CH1OF	0	R	**Channel 1 Data Overflow Interrupt Flag**
	Indicates channel 1 data overflow.			
2	CH0OF	0	R	**Channel 0 Data Overflow Interrupt Flag**
	Indicates channel 0 data overflow.			
1	CH1CD	0	R	**Channel 1 Conversion Done Interrupt Flag**
	Indicates channel 1 conversion complete.			
0	CH0CD	0	R	**Channel 0 Conversion Done Interrupt Flag**
	Indicates channel 0 conversion complete.			

25.5.7 VDACn_IFS - Interrupt Flag Set 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	0	0	0	0					0	0	0	0	0	0	0	0	0										0	0	0	0	0	0
Access	W1	W1	W1	W1					W1	W1	W1	W1	W1	W1	W1	W1	W1										W1	W1	W1	W1	W1	W1
Name	OPA3OUTVALID	OPA2OUTVALID	OPA1OUTVALID	OPA0OUTVALID					OPA3PRSTIMEDERR	OPA2PRSTIMEDERR	OPA1PRSTIMEDERR	OPA0PRSTIMEDERR	OPA3APORTCONFLICT	OPA2APORTCONFLICT	OPA1APORTCONFLICT	OPA0APORTCONFLICT	EM23ERR										CH1UF	CH0UF	CH1OF	CH0OF	CH1CD	CH0CD

Bit	Name	Reset	Access	Description
31	OPA3OUTVALID	0	W1	**Set OPA3OUTVALID Interrupt Flag**
	Write 1 to set the OPA3OUTVALID interrupt flag			
30	OPA2OUTVALID	0	W1	**Set OPA2OUTVALID Interrupt Flag**
	Write 1 to set the OPA2OUTVALID interrupt flag			
29	OPA1OUTVALID	0	W1	**Set OPA1OUTVALID Interrupt Flag**
	Write 1 to set the OPA1OUTVALID interrupt flag			
28	OPA0OUTVALID	0	W1	**Set OPA0OUTVALID Interrupt Flag**
	Write 1 to set the OPA0OUTVALID interrupt flag			
27:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23	OPA3PRSTIMEDERR	0	W1	**Set OPA3PRSTIMEDERR Interrupt Flag**
	Write 1 to set the OPA3PRSTIMEDERR interrupt flag			
22	OPA2PRSTIMEDERR	0	W1	**Set OPA2PRSTIMEDERR Interrupt Flag**
	Write 1 to set the OPA2PRSTIMEDERR interrupt flag			
21	OPA1PRSTIMEDERR	0	W1	**Set OPA1PRSTIMEDERR Interrupt Flag**
	Write 1 to set the OPA1PRSTIMEDERR interrupt flag			
20	OPA0PRSTIMEDERR	0	W1	**Set OPA0PRSTIMEDERR Interrupt Flag**
	Write 1 to set the OPA0PRSTIMEDERR interrupt flag			
19	OPA3APORTCONFLICT	0	W1	**Set OPA3APORTCONFLICT Interrupt Flag**
	Write 1 to set the OPA3APORTCONFLICT interrupt flag			
18	OPA2APORTCONFLICT	0	W1	**Set OPA2APORTCONFLICT Interrupt Flag**
	Write 1 to set the OPA2APORTCONFLICT interrupt flag			

Bit	Name	Reset	Access	Description
17	OPA1APORTCON-FLICT	0	W1	**Set OPA1APORTCONFLICT Interrupt Flag**
	Write 1 to set the OPA1APORTCONFLICT interrupt flag			
16	OPA0APORTCON-FLICT	0	W1	**Set OPA0APORTCONFLICT Interrupt Flag**
	Write 1 to set the OPA0APORTCONFLICT interrupt flag			
15	EM23ERR	0	W1	**Set EM23ERR Interrupt Flag**
	Write 1 to set the EM23ERR interrupt flag			
14:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CH1UF	0	W1	**Set CH1UF Interrupt Flag**
	Write 1 to set the CH1UF interrupt flag			
4	CH0UF	0	W1	**Set CH0UF Interrupt Flag**
	Write 1 to set the CH0UF interrupt flag			
3	CH1OF	0	W1	**Set CH1OF Interrupt Flag**
	Write 1 to set the CH1OF interrupt flag			
2	CH0OF	0	W1	**Set CH0OF Interrupt Flag**
	Write 1 to set the CH0OF interrupt flag			
1	CH1CD	0	W1	**Set CH1CD Interrupt Flag**
	Write 1 to set the CH1CD interrupt flag			
0	CH0CD	0	W1	**Set CH0CD Interrupt Flag**
	Write 1 to set the CH0CD interrupt flag			

25.5.8 VDACn_IFC - Interrupt Flag Clear 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	0	0	0	0					0	0	0	0	0	0	0	0	0										0	0	0	0	0	0
Access	(R)W1	(R)W1	(R)W1	(R)W1					(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1										(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name	OPA3OUTVALID	OPA2OUTVALID	OPA1OUTVALID	OPA0OUTVALID					OPA3PRSTIMEDERR	OPA2PRSTIMEDERR	OPA1PRSTIMEDERR	OPA0PRSTIMEDERR	OPA3APORTCONFLICT	OPA2APORTCONFLICT	OPA1APORTCONFLICT	OPA0APORTCONFLICT	EM23ERR										CH1UF	CH0UF	CH1OF	CH0OF	CH1CD	CH0CD

Bit	Name	Reset	Access	Description
31	OPA3OUTVALID	0	(R)W1	**Clear OPA3OUTVALID Interrupt Flag**
	Write 1 to clear the OPA3OUTVALID interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
30	OPA2OUTVALID	0	(R)W1	**Clear OPA2OUTVALID Interrupt Flag**
	Write 1 to clear the OPA2OUTVALID interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
29	OPA1OUTVALID	0	(R)W1	**Clear OPA1OUTVALID Interrupt Flag**
	Write 1 to clear the OPA1OUTVALID interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
28	OPA0OUTVALID	0	(R)W1	**Clear OPA0OUTVALID Interrupt Flag**
	Write 1 to clear the OPA0OUTVALID interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
27:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23	OPA3PRSTIMEDERR	0	(R)W1	**Clear OPA3PRSTIMEDERR Interrupt Flag**
	Write 1 to clear the OPA3PRSTIMEDERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
22	OPA2PRSTIMEDERR	0	(R)W1	**Clear OPA2PRSTIMEDERR Interrupt Flag**
	Write 1 to clear the OPA2PRSTIMEDERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
21	OPA1PRSTIMEDERR	0	(R)W1	**Clear OPA1PRSTIMEDERR Interrupt Flag**
	Write 1 to clear the OPA1PRSTIMEDERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
20	OPA0PRSTIMEDERR	0	(R)W1	**Clear OPA0PRSTIMEDERR Interrupt Flag**
	Write 1 to clear the OPA0PRSTIMEDERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
19	OPA3APORTCONFLICT	0	(R)W1	**Clear OPA3APORTCONFLICT Interrupt Flag** Write 1 to clear the OPA3APORTCONFLICT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
18	OPA2APORTCONFLICT	0	(R)W1	**Clear OPA2APORTCONFLICT Interrupt Flag** Write 1 to clear the OPA2APORTCONFLICT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
17	OPA1APORTCONFLICT	0	(R)W1	**Clear OPA1APORTCONFLICT Interrupt Flag** Write 1 to clear the OPA1APORTCONFLICT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
16	OPA0APORTCONFLICT	0	(R)W1	**Clear OPA0APORTCONFLICT Interrupt Flag** Write 1 to clear the OPA0APORTCONFLICT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
15	EM23ERR	0	(R)W1	**Clear EM23ERR Interrupt Flag** Write 1 to clear the EM23ERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
14:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	CH1UF	0	(R)W1	**Clear CH1UF Interrupt Flag** Write 1 to clear the CH1UF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
4	CH0UF	0	(R)W1	**Clear CH0UF Interrupt Flag** Write 1 to clear the CH0UF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
3	CH1OF	0	(R)W1	**Clear CH1OF Interrupt Flag** Write 1 to clear the CH1OF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
2	CH0OF	0	(R)W1	**Clear CH0OF Interrupt Flag** Write 1 to clear the CH0OF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
1	CH1CD	0	(R)W1	**Clear CH1CD Interrupt Flag** Write 1 to clear the CH1CD interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).
0	CH0CD	0	(R)W1	**Clear CH0CD Interrupt Flag** Write 1 to clear the CH0CD interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).

25.5.9 VDACn_IEN - Interrupt Enable 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	0	0	0	0					0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0
Access	RW	RW	RW	RW					RW	RW	RW	RW	RW	RW	RW	RW	RW								RW	RW	RW	RW	RW	RW	RW	RW
Name	OPA3OUTVALID	OPA2OUTVALID	OPA1OUTVALID	OPA0OUTVALID					OPA3PRSTIMEDERR	OPA2PRSTIMEDERR	OPA1PRSTIMEDERR	OPA0PRSTIMEDERR	OPA3APORTCONFLICT	OPA2APORTCONFLICT	OPA1APORTCONFLICT	OPA0APORTCONFLICT	EM23ERR								CH1BL	CH0BL	CH1UF	CH0UF	CH1OF	CH0OF	CH1CD	CH0CD

Bit	Name	Reset	Access	Description
31	OPA3OUTVALID	0	RW	**OPA3OUTVALID Interrupt Enable**
	Enable/disable the OPA3OUTVALID interrupt			
30	OPA2OUTVALID	0	RW	**OPA2OUTVALID Interrupt Enable**
	Enable/disable the OPA2OUTVALID interrupt			
29	OPA1OUTVALID	0	RW	**OPA1OUTVALID Interrupt Enable**
	Enable/disable the OPA1OUTVALID interrupt			
28	OPA0OUTVALID	0	RW	**OPA0OUTVALID Interrupt Enable**
	Enable/disable the OPA0OUTVALID interrupt			
27:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23	OPA3PRSTIMEDERR	0	RW	**OPA3PRSTIMEDERR Interrupt Enable**
	Enable/disable the OPA3PRSTIMEDERR interrupt			
22	OPA2PRSTIMEDERR	0	RW	**OPA2PRSTIMEDERR Interrupt Enable**
	Enable/disable the OPA2PRSTIMEDERR interrupt			
21	OPA1PRSTIMEDERR	0	RW	**OPA1PRSTIMEDERR Interrupt Enable**
	Enable/disable the OPA1PRSTIMEDERR interrupt			
20	OPA0PRSTIMEDERR	0	RW	**OPA0PRSTIMEDERR Interrupt Enable**
	Enable/disable the OPA0PRSTIMEDERR interrupt			
19	OPA3APORTCONFLICT	0	RW	**OPA3APORTCONFLICT Interrupt Enable**
	Enable/disable the OPA3APORTCONFLICT interrupt			
18	OPA2APORTCONFLICT	0	RW	**OPA2APORTCONFLICT Interrupt Enable**
	Enable/disable the OPA2APORTCONFLICT interrupt			

Bit	Name	Reset	Access	Description
17	OPA1APORTCON-FLICT	0	RW	**OPA1APORTCONFLICT Interrupt Enable**
	Enable/disable the OPA1APORTCONFLICT interrupt			
16	OPA0APORTCON-FLICT	0	RW	**OPA0APORTCONFLICT Interrupt Enable**
	Enable/disable the OPA0APORTCONFLICT interrupt			
15	EM23ERR	0	RW	**EM23ERR Interrupt Enable**
	Enable/disable the EM23ERR interrupt			
14:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	CH1BL	0	RW	**CH1BL Interrupt Enable**
	Enable/disable the CH1BL interrupt			
6	CH0BL	0	RW	**CH0BL Interrupt Enable**
	Enable/disable the CH0BL interrupt			
5	CH1UF	0	RW	**CH1UF Interrupt Enable**
	Enable/disable the CH1UF interrupt			
4	CH0UF	0	RW	**CH0UF Interrupt Enable**
	Enable/disable the CH0UF interrupt			
3	CH1OF	0	RW	**CH1OF Interrupt Enable**
	Enable/disable the CH1OF interrupt			
2	CH0OF	0	RW	**CH0OF Interrupt Enable**
	Enable/disable the CH0OF interrupt			
1	CH1CD	0	RW	**CH1CD Interrupt Enable**
	Enable/disable the CH1CD interrupt			
0	CH0CD	0	RW	**CH0CD Interrupt Enable**
	Enable/disable the CH0CD interrupt			

25.5.10 VDACn_CH0DATA - Channel 0 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																					0x800											
Access																					RWH											
Name																					DATA											

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	DATA	0x800	RWH	**Channel 0 Data**
	This register contains the value which will be converted by DAC channel 0.			

25.5.11 VDACn_CH1DATA - Channel 1 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																					0x800											
Access																					RWH											
Name																					DATA											

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	DATA	0x800	RWH	**Channel 1 Data**
	This register contains the value which will be converted by DAC channel 1.			

25.5.12 VDACn_COMBDATA - Combined 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					0x800																0x800											
Access					W																W											
Name					CH1DATA																CH0DATA											

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	CH1DATA	0x800	W	**Channel 1 Data**
	Data written to this register will be written to DATA in VDACn_CH1DATA.			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	CH0DATA	0x800	W	**Channel 0 Data**
	Data written to this register will be written to DATA in VDACn_CH0DATA.			

25.5.13 VDACn_CAL - Calibration 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													0x8						0x20											0x4		
Access													RW						RW											RW		
Name													GAINERRTRIMCH1						GAINERRTRIM											OFFSETTRIM		

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	GAINERRTRIMCH1	0x8	RW	**Gain Error Trim Value for CH1**
	This register contains the fine gain error trim for CH1. Program with Device Information value found in DEVINFO_VDACnCH1CAL depending on chosen reference.			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	GAINERRTRIM	0x20	RW	**Gain Error Trim Value**
	This register contains the fine gain error trim for CH0 and coarse gain error trim for CH1. Program with Device Information value found in DEVINFO_VDACnMAINCAL or DEVINFO_VDACnALTCAL depending on chosen reference and choice of main versus alternative output usage.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	OFFSETTRIM	0x4	RW	**Input Buffer Offset Calibration Value**
	This register contains the DAC input buffer offset calibration value. Program with Device Information value found in DEVINFO_VDACnCH1CAL.			

25.5.14 VDACn_OPAx_APORTREQ - Operational Amplifier APORT Request Status Register

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																							0	0	0	0	0	0	0	0		
Access																							R	R	R	R	R	R	R	R		
Name																							APORT4YREQ	APORT4XREQ	APORT3YREQ	APORT3XREQ	APORT2YREQ	APORT2XREQ	APORT1YREQ	APORT1XREQ		

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	APORT4YREQ	0	R	**1 If the Bus Connected to APORT4Y is Requested**
	Reports if the bus connected to APORT4Y is being requested from the APORT			
8	APORT4XREQ	0	R	**1 If the Bus Connected to APORT4X is Requested**
	Reports if the bus connected to APORT4X is being requested from the APORT			
7	APORT3YREQ	0	R	**1 If the Bus Connected to APORT3Y is Requested**
	Reports if the bus connected to APORT3Y is being requested from the APORT			
6	APORT3XREQ	0	R	**1 If the Bus Connected to APORT3X is Requested**
	Reports if the bus connected to APORT3X is being requested from the APORT			
5	APORT2YREQ	0	R	**1 If the Bus Connected to APORT2Y is Requested**
	Reports if the bus connected to APORT2Y is being requested from the APORT			
4	APORT2XREQ	0	R	**1 If the Bus Connected to APORT2X is Requested**
	Reports if the bus connected to APORT2X is being requested from the APORT			
3	APORT1YREQ	0	R	**1 If the Bus Connected to APORT1X is Requested**
	Reports if the bus connected to APORT1X is being requested from the APORT			
2	APORT1XREQ	0	R	**1 If the Bus Connected to APORT2X is Requested**
	Reports if the bus connected to APORT2X is being requested from the APORT			
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

25.5.15 VDACn_OPAx_APORTCONFLICT - Operational Amplifier APORT Conflict Status Register

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																							0	0	0	0	0	0	0	0		
Access																							R	R	R	R	R	R	R	R		
Name																							APORT4YCONFLICT	APORT4XCONFLICT	APORT3YCONFLICT	APORT3XCONFLICT	APORT2YCONFLICT	APORT2XCONFLICT	APORT1YCONFLICT	APORT1XCONFLICT		

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	APORT4YCONFLICT	0	R	**1 If the Bus Connected to APORT4Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT4Y is is also being requested by another peripheral
8	APORT4XCONFLICT	0	R	**1 If the Bus Connected to APORT4X is in Conflict With Another Peripheral** Reports if the bus connected to APORT4X is is also being requested by another peripheral
7	APORT3YCONFLICT	0	R	**1 If the Bus Connected to APORT3Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT3Y is is also being requested by another peripheral
6	APORT3XCONFLICT	0	R	**1 If the Bus Connected to APORT3X is in Conflict With Another Peripheral** Reports if the bus connected to APORT3X is is also being requested by another peripheral
5	APORT2YCONFLICT	0	R	**1 If the Bus Connected to APORT2Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT2Y is is also being requested by another peripheral
4	APORT2XCONFLICT	0	R	**1 If the Bus Connected to APORT2X is in Conflict With Another Peripheral** Reports if the bus connected to APORT2X is is also being requested by another peripheral
3	APORT1YCONFLICT	0	R	**1 If the Bus Connected to APORT1X is in Conflict With Another Peripheral** Reports if the bus connected to APORT1X is is also being requested by another peripheral
2	APORT1XCONFLICT	0	R	**1 If the Bus Connected to APORT1X is in Conflict With Another Peripheral** Reports if the bus connected to APORT1X is is also being requested by another peripheral
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

25.5.16 VDACn_OPAx_CTRL - Operational Amplifier Control Register

Offset	Bit Position																													
0x0A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0	0				0			0x0				0	0				0	1	1	0x2	
Access											RW	RW				RW			RW				RW	RW				RW	RW	RW	RW	
Name											APORTYMASTERDIS	APORTXMASTERDIS				PRSOUTMODE			PRSSEL				PRSMODE	PRSEN				OUTSCALE	HCMDIS	INCBW	DRIVESTRENGTH	

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	APORTYMASTERDIS	0	RW	**APORT Bus Master Disable**

Determines if the OPAx will request the APORT bus Y with POSSEL, NEGSEL or APORTOUTSEL. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously by allowing the OPAx to not master the selected bus. When 1, the determination is expected to be from another peripheral, and the OPAx only passively looks at the bus. When 1, the selection of channel for a selected bus is ignored (the bus is not), and is whatever selection the external device mastering the bus has configured for the APORT bus.

Value	Description
0	Bus mastering enabled
1	Bus mastering disabled

20	APORTXMASTERDIS	0	RW	**APORT Bus Master Disable**

Determines if the OPAx will request the APORT bus X with POSSEL , NEGSEL or APORTOUTSEL. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously by allowing the OPAx to not master the selected bus. When 1, the determination is expected to be from another peripheral, and the OPAx only passively looks at the bus. When 1, the selection of channel for a selected bus is ignored (the bus is not), and is whatever selection the external device mastering the bus has configured for the APORT bus.

Value	Description
0	Bus mastering enabled
1	Bus mastering disabled

19:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	PRSOUTMODE	0	RW	**OPAx PRS Output Select**

Selects OPAx Output to PRS.

Value	Mode	Description
0	WARM	Warm status available on PRS. Warm status indicates that opamp is warm and output is enabled.

Bit	Name	Reset	Access	Description
	1	OUTVALID		Outvalid status available on PRS. Outvalid status indicates that opamp output is settled externally at the load.
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:10	PRSSEL	0x0	RW	**OPAx PRS Trigger Select**

Select Channel 0 PRS input channel.

Value	Mode	Description
0	PRSCH0	PRS ch 0 triggers OPA.
1	PRSCH1	PRS ch 1 triggers OPA.
2	PRSCH2	PRS ch 2 triggers OPA.
3	PRSCH3	PRS ch 3 triggers OPA.
4	PRSCH4	PRS ch 4 triggers OPA.
5	PRSCH5	PRS ch 5 triggers OPA.
6	PRSCH6	PRS ch 6 triggers OPA.
7	PRSCH7	PRS ch 7 triggers OPA.
8	PRSCH8	PRS ch 8 triggers OPA.
9	PRSCH9	PRS ch 9 triggers OPA.
10	PRSCH10	PRS ch 10 triggers OPA.
11	PRSCH11	PRS ch 11 triggers OPA.
12	PRSCH12	PRS ch 12 triggers OPA.
13	PRSCH13	PRS ch 13 triggers OPA.
14	PRSCH14	PRS ch 14 triggers OPA.
15	PRSCH15	PRS ch 15 triggers OPA.

9	PRSMODE	0	RW	**OPAx PRS Trigger Mode**

PRS trigger mode of OPA.

Value	Mode	Description
0	PULSED	PULSED trigger is considered a regular asynchronous pulse that starts OPA warmup sequence. The end of warmup sequence is controlled by timeout settings in OPAxTIMER.
1	TIMED	TIMED trigger is considered a pulse long enough to provide OPA warmup sequence. The end of warmup sequence is controlled by negative edge of the pulse.

8	PRSEN	0	RW	**OPAx PRS Trigger Enable**

Select OPAx conversion trigger.

Value	Description
0	OPAx is triggered by OPAxEN
1	OPAx is triggered by PRS input

Bit	Name	Reset	Access	Description
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	OUTSCALE	0	RW	**Scale OPAx Output Driving Strength**
	Use this to scale OPAx output driving strength.			
	Value	Mode		Description
	0	FULL		Select this for full output driving strength.
	1	HALF		Select this for half output driving strength.
3	HCMDIS	1	RW	**High Common Mode Disable**
	Set to disable high common mode. Disables rail-to-rail on input, while output still remains rail-to-rail. The input voltage to the opamp while HCM is disabled is restricted between VSS and VDD-1.2V. Setting this bit improves output linearity when input is low.			
2	INCBW	1	RW	**OPAx Unity Gain Bandwidth Scale**
	Unity gain bandwidth scale.			
	Value			Description
	0			No scaling
	1			When set the unity gain bandwidth will be scaled by factor of 2.5. useful to make OPA operate faster for closed-loop gain setting greater than 3x.
1:0	DRIVESTRENGTH	0x2	RW	**OPAx Operation Mode**
	Selects OPAx operation mode.			
	Value			Description
	0			Lower accuracy with Low drive strength.
	1			Low accuracy with Low drive strength.
	2			High accuracy with High drive strength.
	3			Higher accuracy with High drive strength.

25.5.17 VDACn_OPAx_TIMER - Operational Amplifier Timer Control Register

Offset	Bit Position																													
0x0AC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x001										0x07						0x00					
Access											RW										RW						RW					
Name											SETTLETIME										WARMUPTIME						STARTUPDLY					

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*		
25:16	SETTLETIME	0x001	RW	**OPAx Output Settling Timeout Value**
	Number of clock cycles to drive the output			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	WARMUPTIME	0x07	RW	**OPAx Warmup Time Count Value**
	OPAx warmup timeout value			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	STARTUPDLY	0x00	RW	**OPAx Startup Delay Count Value**
	OPAx startup delay in us. Used only in PRS sample of mode of stand alone opamp.			

25.5.18 VDACn_OPAx_MUX - Operational Amplifier Mux Configuration Register

Offset	Bit Position																															
0x0B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset						0x0						1		0x6			0xF2								0xF1							
Access						RW						RW		RW			RW								RW							
Name						RESSEL						GAIN3X		RESINMUX			NEGSEL								POSSEL							

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	RESSEL	0x0	RW	**OPAx Resistor Ladder Select**

Configures the resistor ladder tap for OPAx.

Value	Mode	Resistor Value
0	RES0	R2 = 1/3 x R1
1	RES1	R2 = R1
2	RES2	R2 = 1 2/3 x R1
3	RES3	R2 = 2 1/5 x R1
4	RES4	R2 = 3 x R1
5	RES5	R2 = 4 1/3 x R1
6	RES6	R2 = 7 x R1
7	RES7	R2 = 15 x R1

23:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	GAIN3X	1	RW	**OPAx Dedicated 3x Gain Resistor Ladder**

Selects gain of 3x.

Value	Description
0	Disables 3x gain ladder.
1	Enables and sets the gain to 3x. If this is set to 1, RESSEL will only be used externally by other opamps. By default this is set to 1 for dac to work properly. For stand alone opamp and to configure gain based on RESSEL value, users need to configure this to 0.

19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	RESINMUX	0x6	RW	**OPAx Resistor Ladder Input Mux**

These bits selects the source for the input mux to the resistor ladder

Value	Mode	Description
0	DISABLE	Set for Unity Gain

Bit	Name	Reset	Access	Description
	1	OPANEXT		Set for NEXTOUT(x-1) input
	2	NEGPAD		NEG pad connected
	3	POSPAD		POS pad connected
	4	COMPAD		Neg pad of OPA0 connected. Direct input to support common reference.
	5	CENTER		OPA0 and OPA1 Resmux connected to form fully differential instrumentation amplifier.
	6	VSS		VSS connected
15:8	NEGSEL	0xF2	RW	**OPAx Inverting Input Mux**

These bits selects the source for the inverting input on OPAx

Mode	Value	Description
APORT1YCH1	48	Select APORT1YCH1
APORT1YCH3	49	Select APORT1YCH3
APORT1YCH5	50	Select APORT1YCH5
...	...	.....
APORT1YCH31	63	Select APORT1YCH31
APORT2YCH0	80	Select APORT2YCH0
APORT2YCH2	81	Select APORT2YCH2
APORT2YCH4	82	Select APORT2YCH3
...	...	.....
APORT2YCH30	95	Select APORT2YCH30
APORT3YCH1	112	Select APORT3YCH1
APORT3YCH3	113	Select APORT3YCH3
APORT3YCH5	114	Select APORT3YCH5
...	...	.....
APORT3YCH31	127	Select APORT3YCH31
APORT4YCH0	144	Select APORT4YCH0
APORT4YCH2	145	Select APORT4YCH2
APORT4YCH4	146	Select APORT4YCH4
...	...	.....
APORT4YCH30	159	Select APORT4YCH30
DISABLE	240	Input disabled
UG	241	Unity Gain feedback path
OPATAP	242	OPAxTAP as input
NEGPAD	243	Input from NEG PAD

7:0	POSSEL	0xF1	RW	**OPAx Non-inverting Input Mux**

These bits selects the source for the non-inverting input on OPAx

Bit	Name	Reset	Access	Description
	Mode	Value		Description
	APORT1XCH0	32		Select APORT1XCH0
	APORT1XCH2	33		Select APORT1XCH2
	APORT1XCH4	34		Select APORT1XCH4
	...	...		.....
	APORT1XCH30	47		Select APORT1XCH30
	APORT2XCH1	64		Select APORT2XCH1
	APORT2XCH3	65		Select APORT2XCH3
	APORT2XCH5	66		Select APORT2XCH5
	...	...		.....
	APORT2XCH31	79		Select APORT2XCH30
	APORT3XCH0	96		Select APORT3XCH0
	APORT3XCH2	97		Select APORT3XCH2
	APORT3XCH4	98		Select APORT3XCH4
	...	...		.....
	APORT3XCH30	111		Select APORT3XCH30
	APORT4XCH1	128		Select APORT4XCH1
	APORT4XCH3	129		Select APORT4XCH3
	APORT4XCH5	130		Select APORT4XCH5
	...	...		.....
	APORT4XCH31	143		Select APORT4XCH31
	DISABLE	240		Input disabled
	DAC	241		DAC as input
	POSPAD	242		POS PAD as input
	OPANEXT	243		NEXTOUT(x-1) as input. For OPA0 not applicable.
	OPATAP	244		OPAxTAP as input. For OPA2 OPA0TAP.

25.5.19 VDACn_OPAx_OUT - Operational Amplifier Output Configuration Register

Offset	Bit Position																													
0x0B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x00														0x00			0	0	0	1
Access												RW														RW			RW	RW	RW	RW
Name												APORTOUTSEL														ALTOUTPADEN			SHORT	APORTOUTEN	ALTOUTEN	MAINOUTEN

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	APORTOUTSEL	0x00	RW	**OPAx APORT Output**

Select APORT output.

Mode	Value	Description
APORT1YCH1	48	Select APORT1YCH1
APORT1YCH3	49	Select APORT1YCH3
APORT1YCH5	50	Select APORT1YCH5
...	...	.....
APORT1YCH31	63	Select APORT1YCH31
APORT2YCH0	80	Select APORT2YCH0
APORT2YCH2	81	Select APORT2YCH2
APORT2YCH4	82	Select APORT2YCH3
...	...	.....
APORT2YCH30	95	Select APORT2YCH30
APORT3YCH1	112	Select APORT3YCH1
APORT3YCH3	113	Select APORT3YCH3
APORT3YCH5	114	Select APORT3YCH5
...	...	.....
APORT3YCH31	127	Select APORT3YCH31
APORT4YCH0	144	Select APORT4YCH0
APORT4YCH2	145	Select APORT4YCH2
APORT4YCH4	146	Select APORT4YCH4
...	...	.....
APORT4YCH30	159	Select APORT4YCH30

15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
8:4	ALTOUTPADEN	0x00	RW	**OPAx Output Enable Value**
	Set to enable output, clear to disable output			

OUT ENABLE	VALUE	Description
OUT0	xxxx1	Alternate Output 0
OUT1	xxx1x	Alternate Output 1
OUT2	xx1xx	Alternate Output 2
OUT3	x1xxx	Alternate Output 3
OUT4	1xxxx	Alternate Output 4

3	SHORT	0	RW	**OPAx Main and Alternative Output Short**
	Set this to short circuit main and alternative outputs. This will keep the outputs shorted even when the VDAC is disabled.			
2	APORTOUTEN	0	RW	**OPAx Aport Output Enable**
	Set this to enable aport output of OPAx.			
1	ALTOUTEN	0	RW	**OPAx Alternative Output Enable**
	Set this to enable alternative output of OPAx.			
0	MAINOUTEN	1	RW	**OPAx Main Output Enable**
	Set this to enable main output of OPAx.			

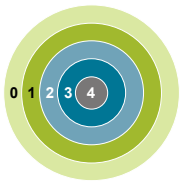
25.5.20 VDACn_OPAx_CAL - Operational Amplifier Calibration Register

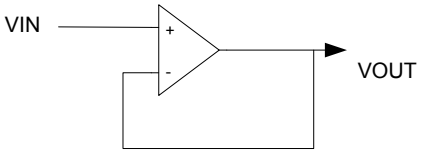
Offset	Bit Position																														
0x0B8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x00						0x00						0x0			0x4				0x0			0x7					0x7			
Access		RW						RW						RW			RW				RW			RW					RW			
Name		OFFSETN						OFFSETP						GM3			GM				CM3			CM2					CM1			

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:26	OFFSETN	0x00	RW	**OPAx Inverting Input Offset Configuration Value**
	This register contains the offset calibration value for inverting input. Program with value obtained from Device Information page (DEVINFO_OPAxCALn) depending on OPAMP number and chosen DRIVESTRENGTH.			
25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24:20	OFFSETP	0x00	RW	**OPAx Non-Inverting Input Offset Configuration Value**
	This register contains the offset calibration value for Non-inverting input offset. Program with value obtained from Device Information page (DEVINFO_OPAxCALn) depending on OPAMP number and chosen DRIVESTRENGTH.			
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:17	GM3	0x0	RW	**Gm3 Trim Value**
	Gm trim code of OPAMP stage 3. Additional trim for OPAMP stage 3. Program with value obtained from Device Information page (DEVINFO_OPAxCALn) depending on OPAMP number and chosen DRIVESTRENGTH.			
16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:13	GM	0x4	RW	**Gm Trim Value**
	Gm trim value common to all OPAMP stages to keep the bandwidth insensitive to process variation. Program with value obtained from Device Information page (DEVINFO_OPAxCALn) depending on OPAMP number and chosen DRIVESTRENGTH.			
12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:10	CM3	0x0	RW	**Compensation Cap Cm3 Trim Value**
	Program with value obtained from Device Information page (DEVINFO_OPAxCALn) depending on OPAMP number and chosen DRIVESTRENGTH.			
9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:5	CM2	0x7	RW	**Compensation Cap Cm2 Trim Value**
	Program with value obtained from Device Information page (DEVINFO_OPAxCALn) depending on OPAMP number and chosen DRIVESTRENGTH.			
4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
3:0	CM1	0x7	RW	**Compensation Cap Cm1 Trim Value** Program with value obtained from Device Information page (DEVINFO_OPAxCALn) depending on OPAMP number and chosen DRIVESTRENGTH.

26. OPAMP - Operational Amplifier





Quick Facts

What?

The opamps are low power amplifiers with a high degree of flexibility targeting a wide variety of standard opamp application areas. With flexible gain and interconnection built-in, they can be configured to support multiple common opamp functions. All pins are available externally for filter configurations. Each opamp has a rail-to-rail input and a rail-to-rail output.

Why?

The opamps are included not only to save energy on a PCB compared to standalone opamps but also to reduce system cost by replacing external opamps.

How?

Two of the opamps are made available as part of the VDAC, while the other opamps are standalone. In addition to popular differential-to-single ended and differential-to-differential driver modes, an ADC unity gain buffer mode configuration makes it possible to isolate kickback noise. The opamps can also be configured as a multi-step cascaded PGA, and for all of the built-in modes no external components are necessary.

26.1 Introduction

The opamps are highly configurable general purpose opamps, suitable for simple filters and buffer applications. The 4 opamps can be configured to support various operational amplifier functions through a network of muxes with possibilities of selecting ranges of on-chip non-inverting and inverting gain configurations and selecting between outputs to various destinations. The opamps can also be configured with external feedback in addition to supporting cascade connections between two or three opamps. The opamps are rail-to-rail in and out. A user selectable mode has been added to optimize linearity, in which case the input voltage to the opamp is restricted to a range between VSS and AVDD-1.2V.

26.2 Features

- 4 individually configurable opamps
- Opamps support rail-to-rail inputs and outputs
- Supports the following functions
  - General opamp mode
  - Voltage follower unity gain
  - Inverting input PGA
  - Non-inverting PGA
  - Cascaded inverting PGA
  - Cascaded non-inverting PGA
  - Two opamp differential amplifier
  - Three opamp differential amplifier
  - Dual buffer ADC driver
- Programmable gain
- Programmable drive strength
- Programmable start delay, warmup and settle time
- Connection to APORT
- Enable / Disable via PRS

- Output status to PRS

26.3 Functional Description

The 4 opamps can be configured to perform various opamp functions through a network of muxes. An overview of the opamps are shown in Figure 26.1 OPAMP System Overview on page 967. Two of the 4 opamps are part of the VDAC, while the others are stand-alone. The outputs of the opamps can be routed to the ADC and ACMP. All 4 opamps can also take input from pins. Since OPA0 and OPA1 are part of the VDAC, special considerations needs to be taken when both VDAC channel 0/channel 1 and OPA0/OPA1 are used. For detailed explanation, refer to 26.3.5 Opamp VDAC Combination.

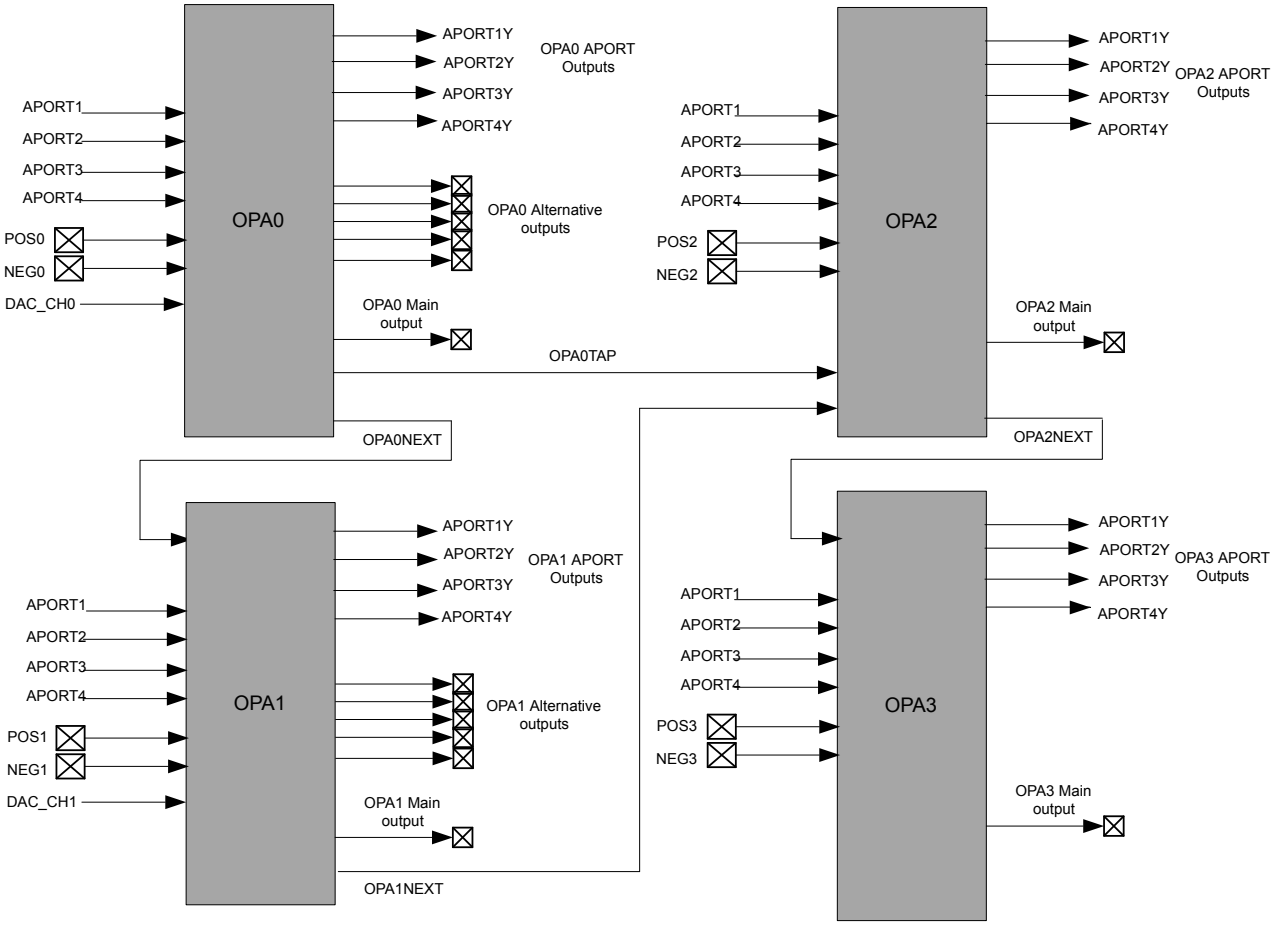


Figure 26.1. OPAMP System Overview

There is a set of input muxes for each opamp, making it possible to select various input sources. A more detailed view of the 4 opamps, including the mux network is shown in Figure 26.2 OPAMP Overview on page 968. The POSSEL mux connected to the positive input makes it possible to select a pin, another opamp output, or tap from the resistor network. Similarly, the NEGSEL mux on the negative input makes it possible to select a pin or a feedback path as its source. The feedback path can be unity gain, 3x gain, or selected from the resistor network for programmable gain. Each opamp has several outputs, a main output, an alternative output network, APORT output and a next output. These outputs make it possible to route the output to a pin, another opamp input, the ADC, the ACMP, or into the feedback path. For details regarding configuring the outputs, see 26.3.1.8 Output Configuration. In addition, there is also a mux to configure the resistor ladder for connection to VSS, a pin, or another opamp output.

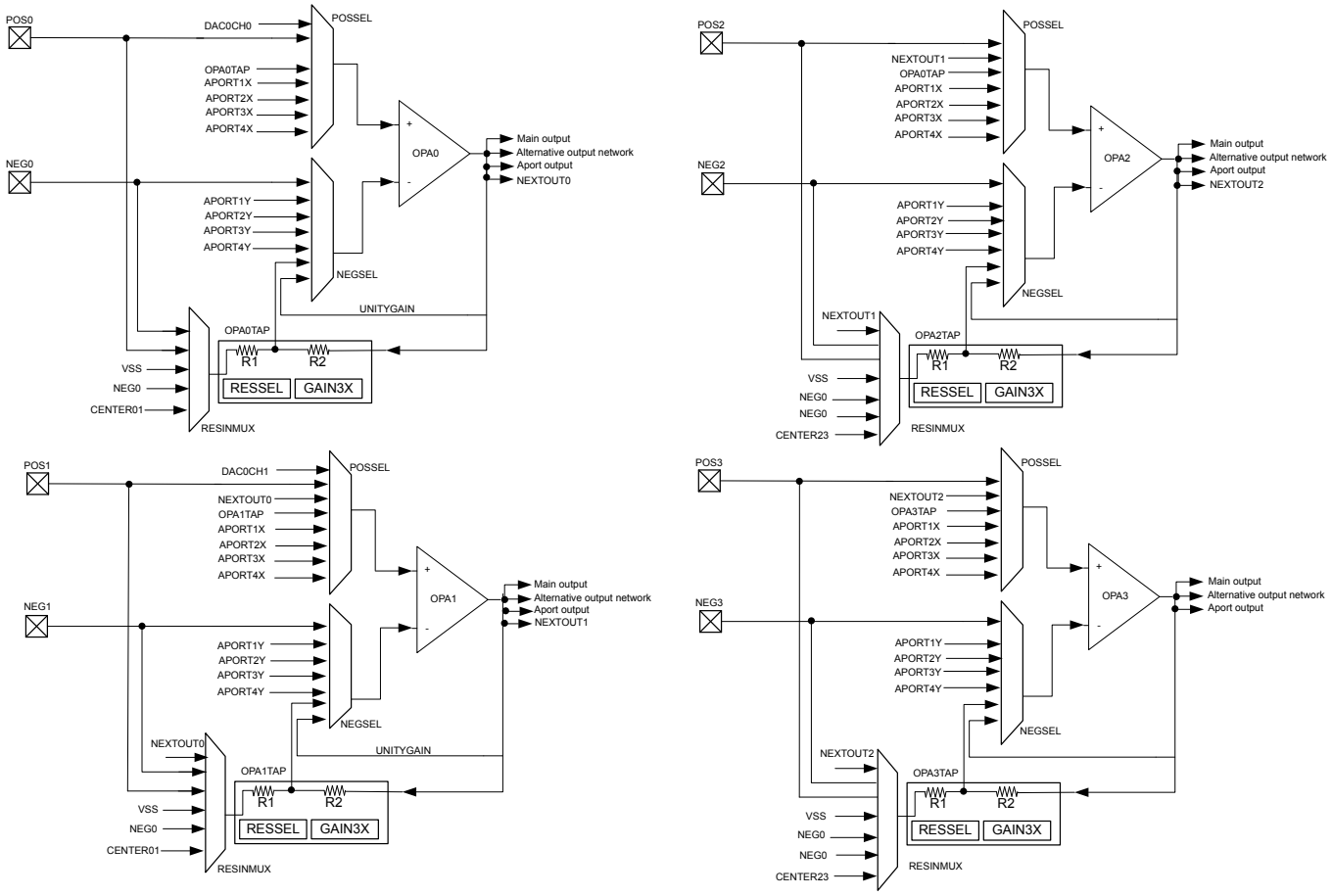


Figure 26.2. OPAMP Overview

26.3.1 Opamp Configuration

Since two of the 4 OPAMPs (OPA0, OPA1) are part of the VDAC, the opamp configuration registers are located in the VDAC.

Each OPAMP can be enabled by setting OPAxEN in VDACn_CMD and can be disabled by setting OPAxDIS in VDACn_CMD. The enabled status of each OPAMP can be read by polling the OPAxENS bit in VDACn_STATUS. OPAxENS goes high immediately after an OPAxEN is written and goes low when OPAxDIS is written and after OPAMP is completely disabled.

Software *must not* write to the following registers while OPAxENS set.

- VDACn_OPAx_CTRL
- VDACn_OPAx_TIMER
- VDACn_OPAx_MUX

26.3.1.1 Enable Sources

Opamp can be enabled either with software or PRS. The default source is software. Setting PRSEN to 1 in VDACn_OPAx_CTRL enables PRS mode. In PRS mode, opamp has two options, which are selectable with PRSMODE in VDACn_OPAx_CTRL. If PRSMODE is configured to TIMED, opamp is turned on on the positive edge of PRS and stays on until PRS goes low. If PRSMODE is configured to PULSED, opamp is turned on the positive edge of PRS and stays on based on the timer configurations in VDACn_OPAxTIMER. The PRS channel is selected by PRSSEL in VDACn_OPAx_CTRL.

26.3.1.2 Warmup Time

When an opamp is enabled some initialization time is required. The warm up period is programmable with WARMUPTIME in VDACn_OPAx_TIME. The OPAxWARM bit in VDACn_STATUS are set when the warmup period has completed.

The warm up period depends on the selected DRIVESTRENGTH in VDACn_OPAx_CTRL.

Table 26.1. OPAMP Warmup Time

DRIVESTRENGTH	WARMUPTIME (µs)
0	100
1	85
2	8
3	6

26.3.1.3 Settle Time

After an opamp is enabled and the warmed-up time has elapsed the output settles externally. The settle period is programmable with SETTLETIME in VDACn_OPAx_TIME. The OPAxOUTVALID bit in VDACn_STATUS is set when the settle period has completed. When in use by the VDAC the default settling time is used.

The settling period depends on the load at opamp output and DRIVESTRENGTH of the opamp. Table 26.2 OPAMP Settling Time on page 969 specifies SETTLETIME settings for a load of 1KOhm and 75pF.

Table 26.2. OPAMP Settling Time

DRIVESTRENGTH	SETTLETIME (µs)
0	60
1	25
2	3
3	1

26.3.1.4 Startup Delay

Each opamp has an option to delay the warm up period. The startup delay is programmable with STARTDLY in VDACn_OPAx_TIME. If STARTDLY is programmed to a non-zero value, the opamp is warmed up after STARTDLY+WARMUPTIME, and the output settles after STARTDLY+WARMUPTIME+SETTLETIME.

26.3.1.5 Power Supply

The opamp module power (V_{OPA}) is derived from the AVDD supply pin.

26.3.1.6 I/O Pin Considerations

The maximum usable analog signal that can be applied to external opamp inputs (or seen on external opamp outputs) depends on several factors: whether the signal is routed through the APORT, whether High Linearity mode is used, whether overvoltage is enabled, and on the IOVDD/AVDD supply voltages, as shown in the Table 26.3 Maximum Usable IO Voltage on page 969 table.

Table 26.3. Maximum Usable IO Voltage

Opamp Pin	Maximum IO Voltage (APORT USED and OVT Enabled/ Disabled)	Maximum IO Voltage (APORT UNUSED, OVT Enabled)	Maximum IO Voltage (APORT UNUSED, OVT Disabled)
Opamp Inputs - Normal Mode	MIN(AVDD, IOVDD)	MIN(AVDD, IOVDD)	MIN(AVDD, IOVDD)
Opamp Inputs - High Linearity Mode	MIN(AVDD - 1.2 V, IOVDD)	MIN(AVDD - 1.2 V, IOVDD)	MIN(AVDD - 1.2 V, IOVDD)
Opamp Outputs	MIN(AVDD, IOVDD)	MIN(AVDD, IOVDD + 2 V)	MIN(AVDD, IOVDD)

26.3.1.7 Input Configuration

The inputs to the opamps are controlled through a set of input muxes. The mux connected to the positive input is configured by the POSSEL bit-field in the VDACn_OPAx_MUX register. Similarly, the mux connected to the negative input is configured by setting the NEGSEL bit-field in VDACn_OPAx_MUX. The input into the resistor ladder can be configured by setting the RESINMUX bit-field in VDACn_OPAx_MUX.

26.3.1.8 Output Configuration

Each opamp has three outputs: the main output, an alternative output network with lower drive strength, and an APORT output with low drive strength. These three outputs can be configured as shown in Figure 26.3 Opamp Output Stage Overview on page 970. The main output can be used to drive the main output by setting MAINOUTEN in VDACn_OPAx_OUT. The alternative output can drive the alternative output network by setting ALTOUTEN in VDACn_OPAx_OUT. The APORT output can drive the APORT selection mux by setting APORTOUTEN in VDACn_OPAx_OUT.

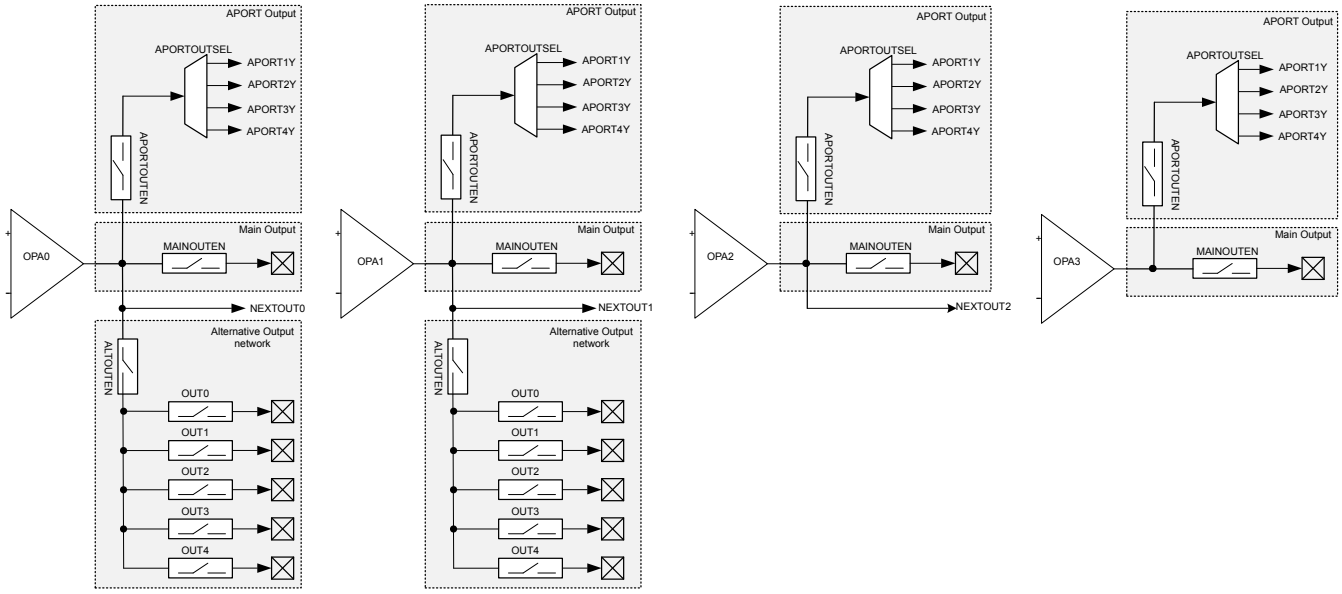


Figure 26.3. Opamp Output Stage Overview

The alternative output network consists of connections to pins and a connection to the next opamp. The connections to pins can be individually enabled by configuring ALTOUTPADEN in te VDACn_OPAx_OUT register. For cascaded opamp configurations, each opamp has a NEXTOUT connection.

The opamp outputs can also be routed to APORT1Y, APORT2Y, APORT3Y, and APORT4Y. The APORT channel can be selected by configuring APORTOUTSEL in VDACn_OPAx_OUT.

The opamps are also routed internally to the ADC. OPA0 and OPA1 are routed through the POSMUX of the ADC, and OPA2 and OPA3 are routed through the NEGMUX of the ADC. See 28.3.7 Input Selection in the ADC chapter for information on how to configure the ADC input mux.

In addition, OPA0 and OPA1 are internally routed to both the POSMUX and NEGMUX of ACMP. See 27.3.6 Input Selection in the ACMP chapter for information on how to configure the ACMP input mux.

The main and alternate outputs of each opamp can be shorted together by setting the SHORT bit-field in VDACn_OPAx_OUT.

26.3.1.9 Gain Programming

The feedback path of each mux includes a resistor ladder that can be used to select a set of gain values. Gain is configured by the RESSEL bit-field located in the VDACn_OPAx_MUX register. Gain values are determined by the resistor ladder based on ratio of R2/R1. It is also possible to bypass the resistor ladder in unity gain mode. In addition, there is also a preconfigured resistor ladder with 3X gain. The 3x gain resistor ladder is enabled by setting GAIN3X in VDACn_OPAx_MUX. By default all opamps are configured in 3x gain mode. When using RESSEL, GAIN3X should be set to zero.

26.3.1.10 Offset Calibration

Each opamp has a calibration register, VDACn_OPAx_CAL, where calibration values for both offset and gain correction can be written. The required calibration settings depend on the chosen DRIVESTRENGTH. The default calibration settings stored in VDACn_OPAx_CAL are for DRIVESTRENGTH=2. If an opamp is being reconfigured, the required calibration settings for DRIVESTRENGTH=n can be found in DEVINFO_OPAxCALn. Offsets can be programmed through the OFFSETP and OFFSETN bitfields of VDACn_OPAx_CAL.

26.3.1.11 Disabling of Rail-to-Rail Operation

Each opamp can have its input rail-to-rail stage disabled by setting the HCMDIS in VDACn_OPAx_CTRL. Disabling the rail-to-rail input stage improves linearity of the opamp, thus improving the total harmonic distortion (THD) at the cost of reduced input signal swing.

26.3.1.12 Unity Gain Bandwidth Scaling

Unity gain bandwidth of an opamp can be scaled setting the INCBW bit in VDACn_OPAx_CTRL. Note that this setting is used only when closed loop gain is greater than 3X. With this setting is enabled, the opamp is not unity gain stable.

26.3.1.13 Opamp Output Scaling

Opamp output drive strength is scaled by one half when the OUTSCALE bit in VDACn_OPAx_CTRL is set.

26.3.2 Interrupts and PRS Output

Each opamp has an interrupt flag OPAxOUTVALID in VDACn_IF that is set when the output is settled externally at the load. An interrupt will be requested if the OPAxOUTVALID interrupt flag in VDACn_IF is set and enabled by the OPAxOUTVALID bit in VDACn_IEN.

The OPAxERRPRSMODE interrupt flag in VDACn_IF indicates a protocol error when the opamp is triggered in PRS TIMED mode. This flag is set if the negative edge of the PRS pulse came before the output to opamp is valid. The interrupt flag is enabled by the OPAxERRPRSMODE bit in VDACn_IEN.

An interrupt can also be requested when an APORT bus conflict occurs if the OPAxAPORTCONFLICT interrupt flag in VDACn_IF is set and enabled through by the OPAxAPORTCONFLICT bit in VDACn_IEN.

One of two aynchronous PRS outputs can be enabled for each opamp by setting PRSOUTMODE in VDACn_OPAx_CTRL. If PRSOUTMODE is WARM, opamp warm-up status is available. If PRSOUTMODE is OUTVALID, opamp output valid status is available.

26.3.3 APORT Request and Conflict Status

The opamps are connected to pins through the APORT system. To help debug over-utilization of APORT resources, the opamps provide request and conflict status information. The request status of APORT buses is visible through the DACn_OPAx_APORTREQ register.

If an APORT bus conflict occurs, it is reported in the DACn_OPAx_APORTCONFLICT register. An APORT conflict occurs if an opamp requests the same bus at the same time as another analog peripheral. In addition an APORT conflict is reported if any two of NEGSEL, POSSEL or APORTOUTSEL are configured to request the same APORT bus.

It is possible for the opamps to passively monitor APORT buses without controlling the switches and creating bus conflicts. This can be done by setting APORTXMASTERDIS or APORTYMASTERDIS in the DACn_OPAx_CTRL register.

26.3.4 Opamp Modes

The opamps can perform several different functions by configuring the internal signal routing between the opamps. The modes available are described in the following sections.

26.3.4.1 General Opamp Mode

In this mode, the resistor ladder is isolated from the feedback path, and the input signal routing is defined by POSSEL and NEGSEL in VDACn_OPAx_MUX. The output signal routing is defined by the setting of VDACn_OPAx_OUT.

Table 26.4. General Opamp Mode Configuration

OPA Bitfields	OPA Configuration
OPAx POSSEL	POSPADx, APORT[1-4]X
OPAx NEGSEL	OPATAP, UG, NEGPADx, APORT[1-4]Y
OPAx RESINMUX	NEXTOUT, POSPADx, NEGPADx, VSS

26.3.4.2 Voltage Follower Unity Gain

In this mode, the unity gain feedback path is selected for the negative input by setting the NEGSEL bit-field to UG in the VDACn_OPAx_MUX register as shown in Figure 26.4 Voltage Follower Unity Gain Overview on page 972. The positive input is selected by the POSSEL bit-field in VDACn_OPAx_MUX, and the output is configured by VDACn_OPAx_OUT register.

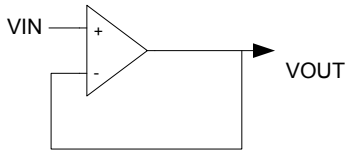


Figure 26.4. Voltage Follower Unity Gain Overview

Table 26.5. Voltage Follower Unity Gain Configuration

OPA Bitfields	OPA Configuration
OPAx POSSEL	OPATAP, NEXTOUT, POSPADx, APORT[1-4]X
OPAx NEGSEL	UG
OPAx RESINMUX	DISABLE

26.3.4.3 Inverting Input PGA

Figure 26.5 Inverting Input PGA Overview on page 972 shows the inverting input PGA configuration. In this mode, the negative input is connected to the resistor ladder by setting the NEGSEL bit-field to OPATAP in the VDACn_OPAx_MUX register. This setting provides a programmable gain on the negative input, which is set by the RESSEL bit-field in VDACn_OPAx_MUX. Signal ground for the positive input can come from off-chip by setting the POSSEL bit-field to PAD or APORT in VDACn_OPAx_MUX. In addition, the output is configured by VDACn_OPAx_OUT register.

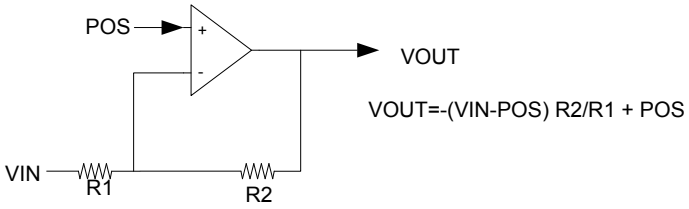


Figure 26.5. Inverting Input PGA Overview

Table 26.6. Inverting Input PGA Configuration

OPA Bitfields	OPA Configuration
OPAx POSSEL	POSPADx, APORT[1-4]X
OPAx NEGSEL	OPATAP
OPAx RESINMUX	NEXTOUT, NEGPADx, POSPADx

26.3.4.4 Non-inverting PGA

Figure 26.6 Non-inverting PGA Overview on page 973 shows the non-inverting input configuration. In this mode, the negative input is connected to the resistor ladder by setting the NEGSEL bit-field to OPATAP in VDACn_OPAx_MUX. This setting provides a programmable gain on the negative input, which is set by the RESSEL bit-field in VDACn_OPAx_MUX. In addition, the RESINMUX bit-field must be set to VSS or NEGPAD in VDACn_OPAx_MUX. The positive input is selected by the POSSEL bit-field, and the output is configured by VDACn_OPAx_OUT register.

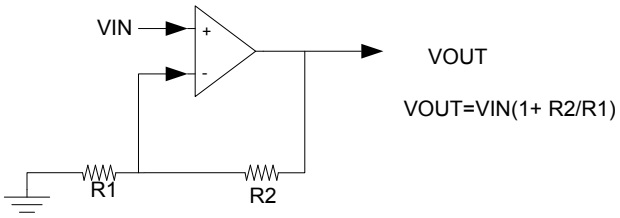


Figure 26.6. Non-inverting PGA Overview

Table 26.7. Non-inverting PGA Configuration

OPA Bitfields	OPA Configuration
OPAx POSSEL	NEXTOUT, POSPADx, APORT[1-4]X
OPAx NEGSEL	OPATAP
OPAx RESINMUX	VSS, NEGPAD

26.3.4.5 Cascaded Inverting PGA

This mode enables the opamp signals to be internally configured to cascade two or more opamps in inverting mode as shown in Figure 26.7 Cascaded Inverting PGA Overview on page 974. In both cases, the positive input is connected to signal ground by setting the POSSEL bit-field to PAD or APORT in VDACn_OPAx_MUX. When cascaded, the negative input is connected to the resistor ladder by setting the NEGSEL bit-field to OPATAP in VDACn_OPAx_MUX. The input to the resistor ladder is configured by the RESINMUX bit-field in VDACn_OPAx_MUX.

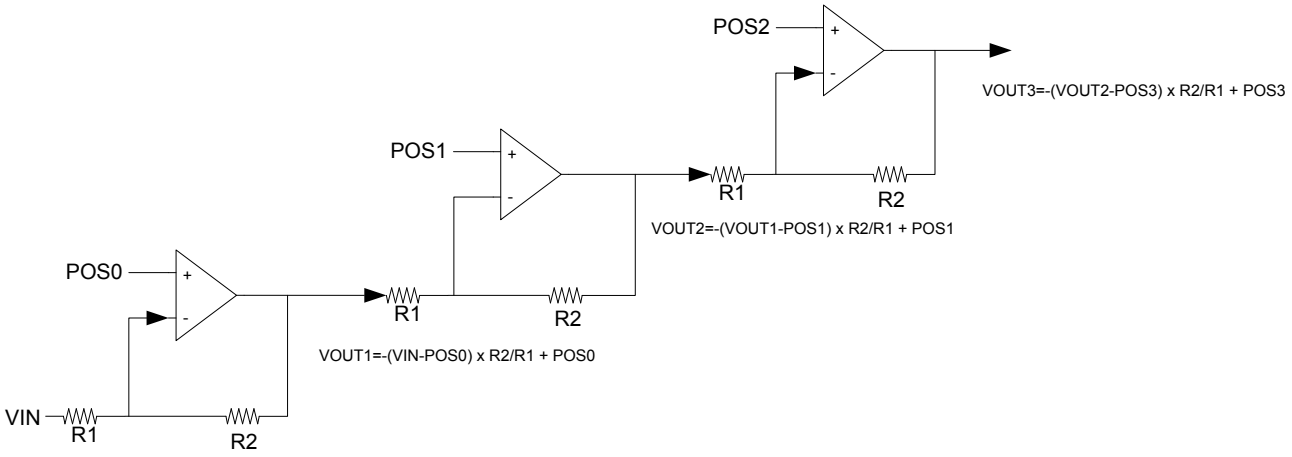


Figure 26.7. Cascaded Inverting PGA Overview

Table 26.8 Cascaded Inverting PGA Configuration on page 974 shows cascaded non-inverting PGA with OPA0,OPA1 and OPA2. The output from OPA0 is connected to OPA1 to create the second stage by setting the RESINMUX field to OPANEXT in VDACn_OPA1_MUX. The last stage is created by setting the RESINMUX bit-field to OPANEXT in VDACn_OPA2MUX.

Table 26.8. Cascaded Inverting PGA Configuration

OPA	OPA Bitfields	OPA Configuration
OPA0	POSSEL	POSPAD0, APORT[1-4]X
OPA0	NEGSEL	OPATAP
OPA0	RESINMUX	NEGPAD0
OPA1	POSSEL	POSPAD1, APORT[1-4]X
OPA1	NEGSEL	OPATAP
OPA1	RESINMUX	OPANEXT
OPA2	POSSEL	POSPAD2,APORT[1-4]X
OPA2	NEGSEL	OPATAP
OPA2	RESINMUX	OPANEXT

26.3.4.6 Cascaded Non-inverting PGA

This mode enables the opamp signals to be internally configured to cascade two or more opamps in non-inverting mode as shown in Figure 26.8 Cascaded Non-inverting PGA Overview on page 975. The negative input for all opamps will be connected to the resistor ladder by setting the NEGSEL bit-field to OPATAP. In addition the resistor ladder input must be set to VSS or NEGPADx by configuring the RESINMUX bit-field in VDACn_OPAx_MUX.

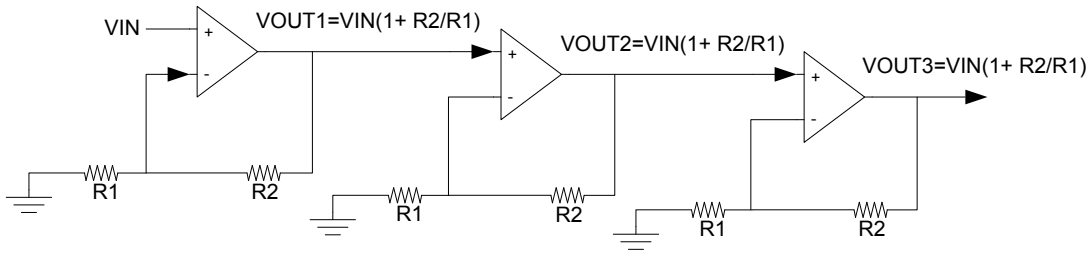


Figure 26.8. Cascaded Non-inverting PGA Overview

Table 26.9 Cascaded Non-inverting PGA Configuration on page 975 shows cascaded non-inverting PGA with OPA0,OPA1 and OPA2. When cascaded, the positive input on OPA0 is configured by the OPA0 POSSEL bit-field in VDACn_OPA0_MUX. The output from OPA0 is connected to OPA1 to create the second stage by setting the POSSEL field to OPANEXT in VDACn_OPA1_MUX. The last stage is created by setting the POSSEL bit-field to OPANEXT in VDACn_OPA2_MUX.

Table 26.9. Cascaded Non-inverting PGA Configuration

OPA	OPA Bitfields	OPA Configuration
OPA0	POSSEL	POSPAD0,APORT[1-4]X
OPA0	NEGSEL	OPATAP
OPA0	RESINMUX	VSS, NEGPAD0
OPA1	POSSEL	OPANEXT
OPA1	NEGSEL	OPATAP
OPA1	RESINMUX	VSS, NEGPAD1
OPA2	POSSEL	OPANEXT
OPA2	NEGSEL	OPATAP
OPA2	RESINMUX	VSS, NEGPAD2

26.3.4.7 Two Opamp Differential Amplifier

This mode allows OPA0 and OPA1 or OPA1 and OPA2 to be internally connected to form a two opamp differential amplifier as shown in Figure 26.9 Two Op-amp Differential Amplifier Overview on page 976. When using OPA0 and OPA1, the positive input of OPA0 can be connected to any input by setting the POSSEL bit-field in VDACn_OPA0_MUX. The OPA0 feedback path must be configured for unity gain by setting the NEGSEL bit-field to UG in VDACn_OPA0_MUX. In addition, the OPA0 RESINMUX bit-field must be set to DISABLED. The OPA0 NEXTOUT output must be connected to OPA1 by setting the RESINMUX bit-field to OPANEXT in VDAC_n_OPA1_MUX. The positive input onof OPA1 is selected by the POSSELbit-field in VDACn_OPA1_MUX. The OPA1 output is configured by DACn_OPA1_OUT.

When using OPA1 and OPA2, the positive input of OPA1 can be connected to any input by setting the POSSEL bit-field in VDACn_OPA1_MUX. The OPA1 feedback path must be configured for unity gain by setting the NEGSEL bit-field to UG in VDACn_OPA1_MUX. In addition, the OPA1 RESINMUX bit-field must be set to DISABLED. The OPA1 NEXTOUT output must be connected to OPA2 by setting the RESINMUX bit-field to OPANEXT in VDACn_OPA2_MUX. The positive input of OPA2 is selected by the POSSEL bit-field in VDACn_OPA2_MUX. The OPA2 output is configured by DACn_OPA2_OUT.

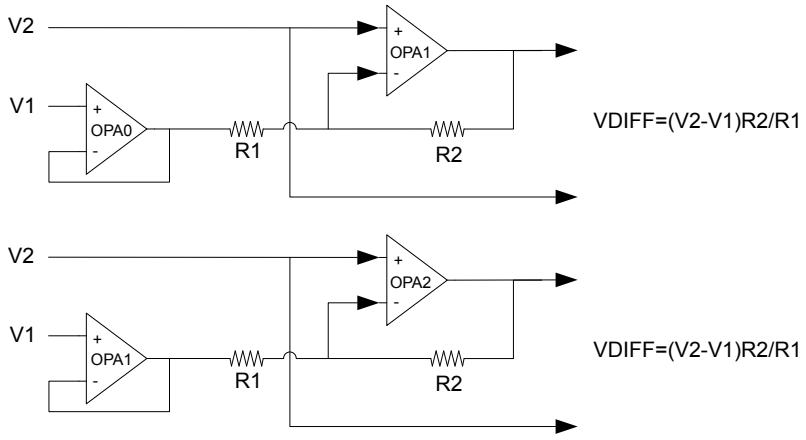


Figure 26.9. Two Op-amp Differential Amplifier Overview

Table 26.10. OPA0/OPA1 Differential Amplifier Configuration

OPA	OPA Bitfields	OPA Configuration
OPA0	POSSEL	POSPAD0, APORT[1-4]X
OPA0	NEGSEL	UG
OPA0	RESINMUX	DISABLE
OPA1	POSSEL	POSPAD1, APORT[1-4]X
OPA1	NEGSEL	OPATAP
OPA1	RESINMUX	OPANEXT

Table 26.11. OPA1/OPA2 Differential Amplifier Configuration

OPA	OPA Bitfields	OPA Configuration
OPA1	POSSEL	POSPAD1, APORT[1-4]X
OPA1	NEGSEL	UG
OPA1	RESINMUX	DISABLE
OPA2	POSSEL	POSPAD2, APORT[1-4]X
OPA2	NEGSEL	OPATAP
OPA2	RESINMUX	OPANEXT

26.3.4.8 Three Opamp Differential Amplifier

This mode allows the three opamps to be internally configured to form a three opamp differential amplifier as shown in Figure 26.10 Three Op-amp Differential Amplifier Overview on page 977. For both OPA0 and OPA1, the positive input can be connected to any input by configuring the OPA0 POSSEL and OPA1 POSSEL bitfields in VDACn_OPA0_MUX and VDACn_OPA1_MUX, respectivley. The OPA0 and OPA1 feedback paths must be configured for unity gain by setting the OPA0 NEGSEL and OPA1 NEGSEL bitfields to UG in VDACn_OPA0_MUX and VDACn_OPA1_MUX respectivley. In addition the OPA0 RESINMUX and OPA1 RESINMUX bitfields must be set to DISABLED. The OPA1 output must be connected to OPA2 by setting RESINMUX to OPANEXT in VDACn_OPA2_MUX and the OPA2 POSSEL must be set to OPATAP. The OPA2 output is configured by the DACn_OPA2_OUT register.

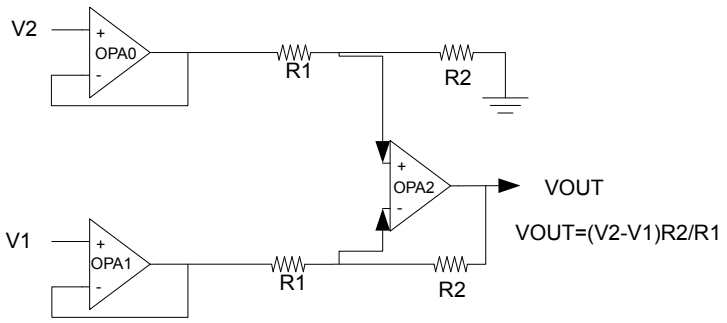


Figure 26.10. Three Op-amp Differential Amplifier Overview

The gain for the Three Opamp Differential Amplifier is determined by the combination of the gain settings of OPA0 and OPA2. Gain values of 1/3, 1 and 3, are available and programmed as shown in the table below.

Table 26.12. Three Opamp Differential Amplifier Gain Programming

Gain	OPA0 RESSEL	OPA2 RESSEL
1/3	4	0
1	1	1
3	0	4

Table 26.13. Three Opamp Differential Amplifier Configuration

OPA	OPA Bitfields	OPA Configuration
OPA0	POSSEL	POSPAD0, APORT[1-4]X
OPA0	NEGSEL	UG
OPA0	RESINMUX	DISABLE
OPA1	POSSEL	POSPAD1, APORT[1-4]X
OPA1	NEGSEL	UG
OPA1	RESINMUX	DISABLE
OPA2	POSSEL	OPATAP
OPA2	NEGSEL	OPATAP
OPA2	RESINMUX	OPANEXT

26.3.4.9 Instrumentation Amplifier

OPA0 and OPA1 can form a fully differential instrumentation amplifier by setting RESINMUX to CENTER for both opamps in VDACn_OPA0_MUX and VDACn_OPA1_MUX. Configuring RESINMUX to CENTER makes a connection between resistor ladder of the opamps as shown in Figure 26.11 Instrumentation Amplifier Overview on page 978.

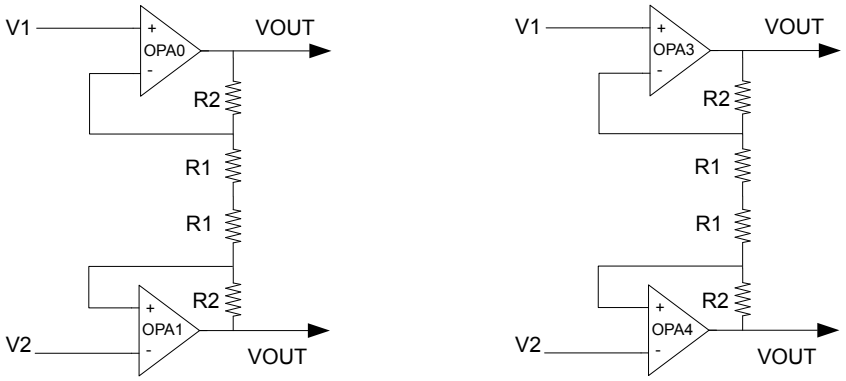


Figure 26.11. Instrumentation Amplifier Overview

26.3.4.10 Common Reference

It is possible to configure all opamps to have a common reference by setting the RESINMUX to COMPAD in VDACn_OPAx_MUX. When RESINMUX of all opamps is set to COMPAD mode, the NEGPAD input of OPA0 is used.

26.3.4.11 Dual Buffer ADC Driver

It is possible to use any two of the opamps to form a Dual Buffer ADC driver as shown in Figure 26.12 Dual Buffer ADC Driver Overview on page 978. Both opamps used must be configured in the same way. The positive input is configured by setting the 0PAx POSSEL to PAD, and the negative input is connected to the resistor ladder by setting NEGSEL to OPATAP in VDACn_OPAx_MUX. The output from the opamps can be configure to drive pins through the alternative output network or the APORT. The ADC can sample pins that the opamps are driving through the APORT.

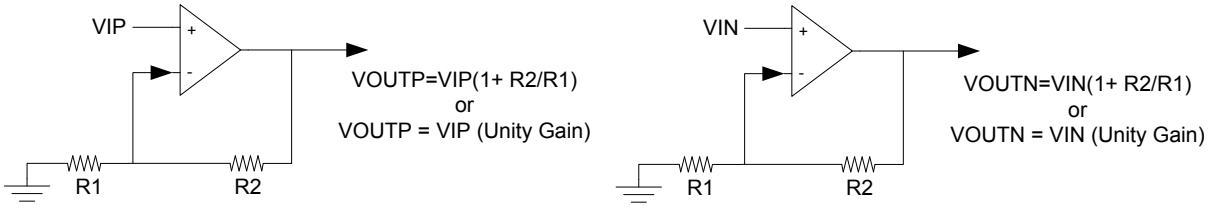


Figure 26.12. Dual Buffer ADC Driver Overview

Table 26.14. Dual Buffer ADC Driver Configuration

OPA	OPA Bitfields	OPA Configuration
OPAx	POSSEL	POSPADx, APORT[1-4]X
OPAx	NEGSEL	OPATAP
OPAx	RESINMUX	VSS

26.3.5 Opamp VDAC Combination

Since two of the OPAMPs are part of the VDAC, it is not possible to use both VDAC channels and all 4 OPAMPs at the same time. If both VDAC channels are used, OPA0 and OPA1 can not be used as stand-alone opamp. However, it is possible to use one of the VDAC channels in combination with OPA0 or OPA1. OPA1 is available when VDAC channel 0 is in use, and OPA0 is available when VDAC channel 1 is used.

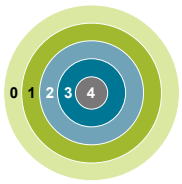
26.4 Register Map

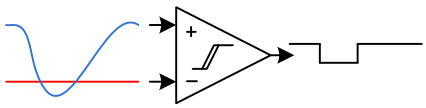
The register map of the opamp can be found in 25.4 Register Map in the VDAC chapter.

26.5 Register Description

The register description of the opamp can be found in 25.5 Register Description in the VDAC chapter.

27. ACMP - Analog Comparator





Quick Facts

What?

The Analog Comparator (ACMP) compares two analog signals and returns a digital value telling which is greater.

Why?

Applications often do not need to know the exact value of an analog signal, only if it has passed a certain threshold. Often the voltage must be monitored continuously, which requires extremely low power consumption.

How?

Available down to Energy Mode 3 and using as little as 100 nA, the ACMP can wake up the system when input signals pass the threshold. The analog comparator can compare two analog signals or one analog signal and a highly configurable internal reference.

27.1 Introduction

The Analog Comparator compares the voltage of two analog inputs and outputs a digital signal indicating which input voltage is higher. Inputs can either be from internal references or from external pins. Response time, and thereby the current consumption, can be configured by altering the current supply to the comparator.

27.2 Features

- Up to 160 selectable external I/O inputs for both positive and negative inputs
  - Up to 48 I/O can be used as a dividable reference
- 6 selectable internal inputs
  - VDAC channel 0 voltage as a reference
  - VDAC channel 1 voltage as a reference
  - 5V regulator output as a reference
  - Dividable Internal 1.25 V bandgap reference voltage
  - Dividable Internal 2.5 V bandgap reference voltage
  - Dividable $V_{ACMPVDD}$ reference voltage
- Voltage supply monitoring
- Low power mode for internal V_{DD} and bandgap references
- Selectable hysteresis
  - 8 values
  - Values can be positive or negative
  - Dividable references have scale for both both output values, allowing for even larger hysteresis
- Selectable response time
- Asynchronous interrupt generation on selectable edges
  - Rising edge
  - Falling edge
  - Both edges
- Operational in EM0 Active down to EM3 Stop
- Dedicated capacitive sense mode with up to 8 inputs
  - Adjustable internal resistor
- Configurable output when inactive
- Comparator output direct on PRS
- Comparator output on GPIO through alternate functionality
  - Output inversion available

27.3 Functional Description

An overview of the ACMP is shown in Figure 27.1 ACMP Overview on page 982 .

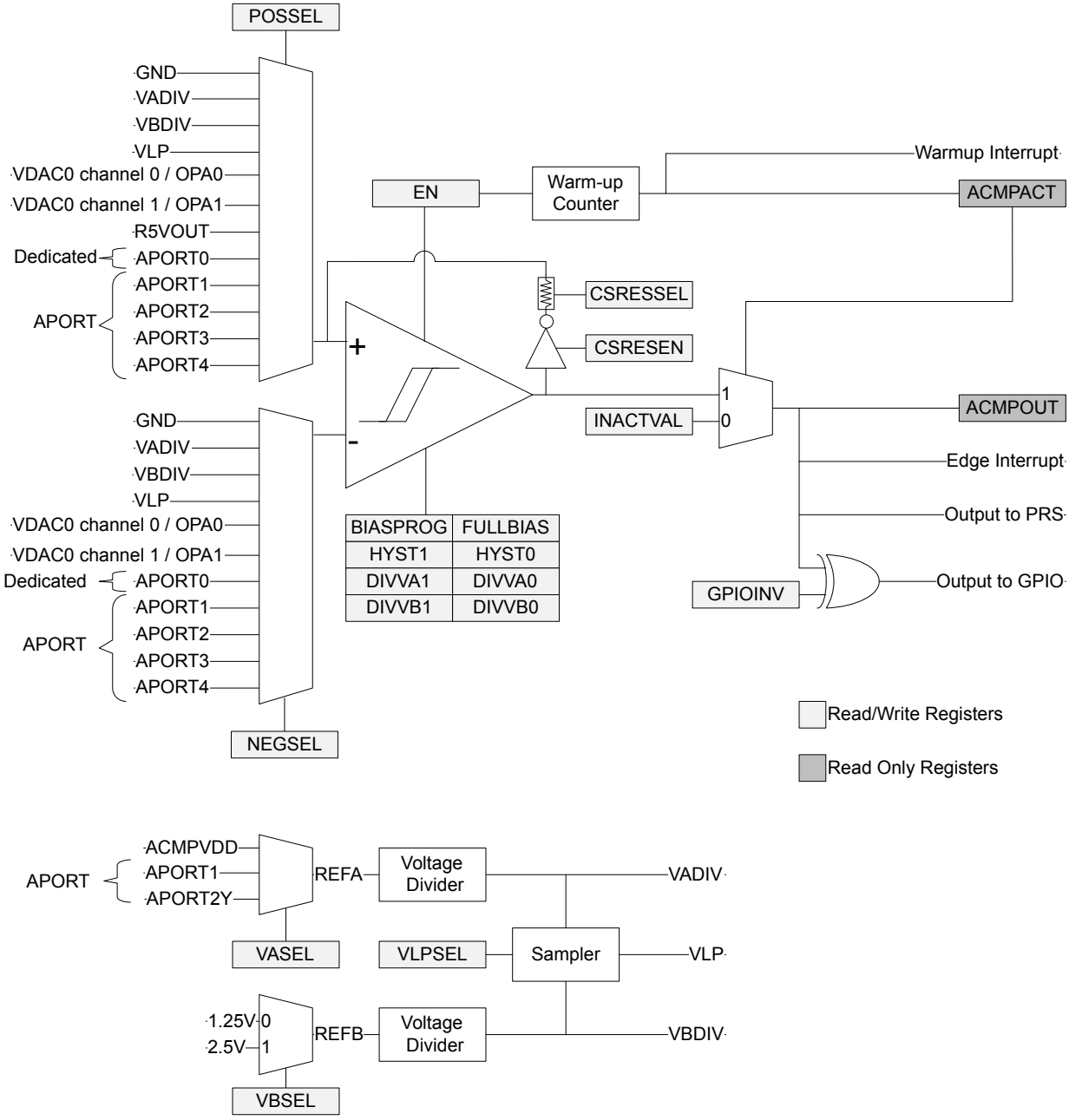


Figure 27.1. ACMP Overview

The comparator has two analog inputs: one positive and one negative. When the comparator is active, the output indicates which of the two input voltages is higher. When the voltage on the positive input is higher than the voltage on the negative input, the digital output is high and vice versa.

The output of the comparator can be read in the ACMPOUT bit in ACMPn_STATUS. It is possible to switch inputs while the comparator is enabled, but all other configuration should only be changed while the comparator is disabled.

27.3.1 Power Supply

The comparator power supply ($V_{ACMPVDD}$) can be configured to be AVDD, DVDD, or IOVDD using the PWRSEL bitfield in ACMPn_CTRL. By default, $V_{ACMPVDD}$ is set to AVDD.

27.3.2 Warm-up Time

The analog comparator is enabled by setting the EN bit in ACMPn_CTRL. The comparator requires some time to stabilize after it is enabled. This time period is called the warm-up time. The warm-up period is self-timed and will complete within 5µs after EN is set.

During warm-up and when the comparator is disabled, the output level of the comparator is set to the value of the INACTVAL bit in ACMPn_CTRL. When the warm-up time is over, the ACMPACT bit in ACMPn_STATUS is set to 1 to indicate that the comparator is active.

An edge interrupt will be generated if the edge interrupt is enabled and the value set in INACTVAL differs from ACMPOUT when the comparator transitions from warm-up to active.

Software should wait until the warm-up period is over before entering EM2 or EM3, otherwise no comparator interrupts will be detected. EM1 can still be entered during warm-up. After the warm-up period is completed, interrupts will be detected in EM2 and EM3.

27.3.3 Response Time

There is a delay from when the input voltage changes polarity to when the output toggles. This delay is called the response time and can be altered by increasing or decreasing the bias current to the comparator through the BIASPROG and FULLBIAS fields in the ACMPn_CTRL register. The current and speed of the circuit increase as the values of FULLBIAS and BIASPROG are increased from their minimum setting of FULLBIAS=0 BIASPROG=0b00000 to the maximum setting FULLBIAS=1 BIASPROG=0b11111 (maximum). The setting of FULLBIAS has a greater affect on current and speed than the setting of BIASPROG. See the part data sheet for specific current and response times related to the setting of these fields.

If FULLBIAS is set, to avoid glitches the highest hysteresis level should be used.

27.3.4 Hysteresis

When the hysteresis level is set to a non-zero value, the digital output will not toggle until the positive input voltage is at a voltage equal to the hysteresis level above or below the negative input voltage (see Figure 27.3 Hysteresis on page 984). This feature can be used to avoid continual comparator output changes due to noise when the positive and negative inputs are nearly equal by requiring the input difference to exceed the hysteresis threshold.

In the analog comparator, hysteresis can be configured to 8 different levels. Level 0 is no hysteresis. Hysteresis is configured through the HYST field in ACMPn_HYSTERESIS0 and ACMPn_HYSTERESIS1 registers. The hysteresis value can be positive or negative. The comparator will output a 1 if:

POSSEL - NEGSEL > HYST

There are two hysteresis registers, ACMPn_HYSTERESIS0 and ACMPn_HYSTERESIS1, as the ACMP supports asymmetric hysteresis. ACMPn_HYSTERESIS0 are the hysteresis values used when the comparator output is 0; ACMPn_HYSTERESIS1 are the values used when the comparator output is 1. The user must set both registers to the same values if symmetric hysteresis is desired.

Along with the HYST field, the ACMPn_HYSTERESIS0/1 registers include the DIVVA and DIVVB fields. This allows the user to implement even larger hysteresis when comparing against VADIV or VBDIV, as the reference voltage can vary with the comparator output, also.

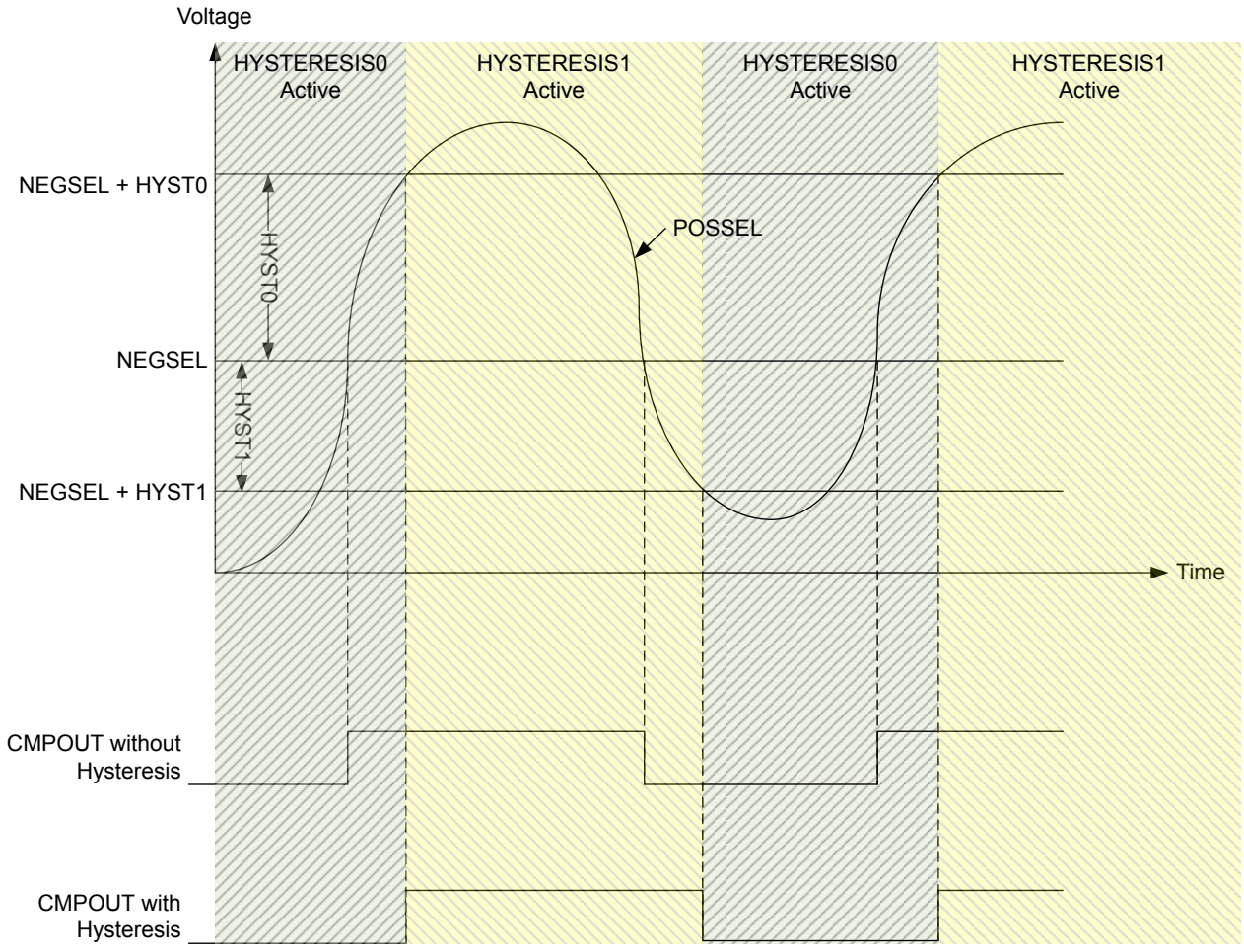


Figure 27.3. Hysteresis

27.3.5 Input Pin Considerations

For external ACMP inputs routed through the APORT, the maximum supported analog input voltage will be limited to the MIN($V_{ACMPVDD}$, IOVDD) (where $V_{ACMPVDD}$ is selected by the PWRSEL bitfield in ACMPn_CTRL). Note that pins configured as ACMP inputs should disable OVT (by setting the corresponding GPIO_Px_OVTDIS bit) to reduce any potential distortion introduced by the OVT circuitry.

27.3.6 Input Selection

The POSSEL and NEGSEL fields in ACMPn_INPUTSEL control the input connections to the positive and negative inputs of the comparator. The user can select external GPIO pins on the chip, or select a number of internal chip voltages. Pins are selected by configuring channels on APORT buses. Not all selectable channels are available on a given device, as different devices within a family may not implement or bring out all of the I/O defined for that family. Refer to the data sheet for channel availability and pin mapping.

There are limitations on the POSSEL and NEGSEL connections that can be made. The user cannot select an X-bus for both POSSEL and NEGSEL simultaneously, nor a Y-bus for both POSSEL and NEGSEL simultaneously. The second limitation is that when using the feedback resistor only X-bus selections can be made for POSSEL. (The resistor only physically exists on the positive input of the comparator).

The user may also select from a number of internal voltages. VADIV and VBDIV are two dividable voltages. VADIV can be $V_{ACMPVDD}$ divided, or the user can choose to select inputs from a number of APORT buses. VBDIV consists of two dividable band-gap references of either 1.25V or 2.5V. Each of these voltages have dividers in the ACMPn_HYSTERESIS0/1 registers. The formula for the division of these voltages is:

$$VADIV = VA \cdot ((DIVVA+1) / 64)$$

Figure 27.3. VA Voltage Division

$$VBDIV = VB \cdot ((DIVVB+1) / 64)$$

Figure 27.4. VB Voltage Division

Either VADIV and VBDIV can also be used as an input to a lower power reference: VLP. Which of the two is used is configured via the VLPSEL field in ACMPn_INPUTSEL. If the user selects VLP as an input source, then VADIV or VBDIV cannot be used as the source for the other input.

Note: The VLP should not be selected as an input source when the external override interface is enabled.

The POSSEL and NEGSEL fields also allow input from the on-chip VDAC channel 0 or VDAC channel 1. The POSSEL field also allow input from 5V regulator.

ACMP can be configured to operate with a selected level of accuracy depending on the setting of ACCURACY in ACMPn_CTRL. The default is low-accuracy mode where ACMP operates with lower accuracy but consumes less current. When higher accuracy is needed the user can set ACCURACY=1 at the cost of higher current consumption.

27.3.7 Capacitive Sense Mode

The analog comparator includes specialized hardware for capacitive sensing of passive push buttons. Such buttons are traces on the PCB laid out in a way that creates a parasitic capacitor between the button and the ground node. Because a human finger will have a small intrinsic capacitance to ground, the capacitance of the button will increase when the button is touched. The capacitance is measured by including the capacitor in a free-running RC oscillator (see Figure 27.5 Capacitive Sensing Setup on page 987). The frequency produced will decrease when the button is touched compared to when it is not touched. By measuring the output frequency with a timer (via the PRS), the change in capacitance can be detected.

The analog comparator contains a feedback loop including an optional internal resistor. This resistor is enabled by setting the CSRESEN bit in ACMPn_INPUTSEL. The resistance can be set to any of 8 values by configuring the CSRESSEL bits in ACMPn_INPUTSEL. The source for VADIV is set to $V_{ACMPVDD}$ by setting field VASEL=0 in ACMPn_INPUTSEL. The oscillation rails are defined by the VADIV fields in registers ACMPn_HYSTERESIS0/1. The user should select VADIV as the source for NEGSEL, and APORTXCHc for POSSEL in ACMPn_INPUTSEL. When enabled, the comparator output will oscillate between the rails defined by VADIV in ACMPn_HYSTERESIS0/1.

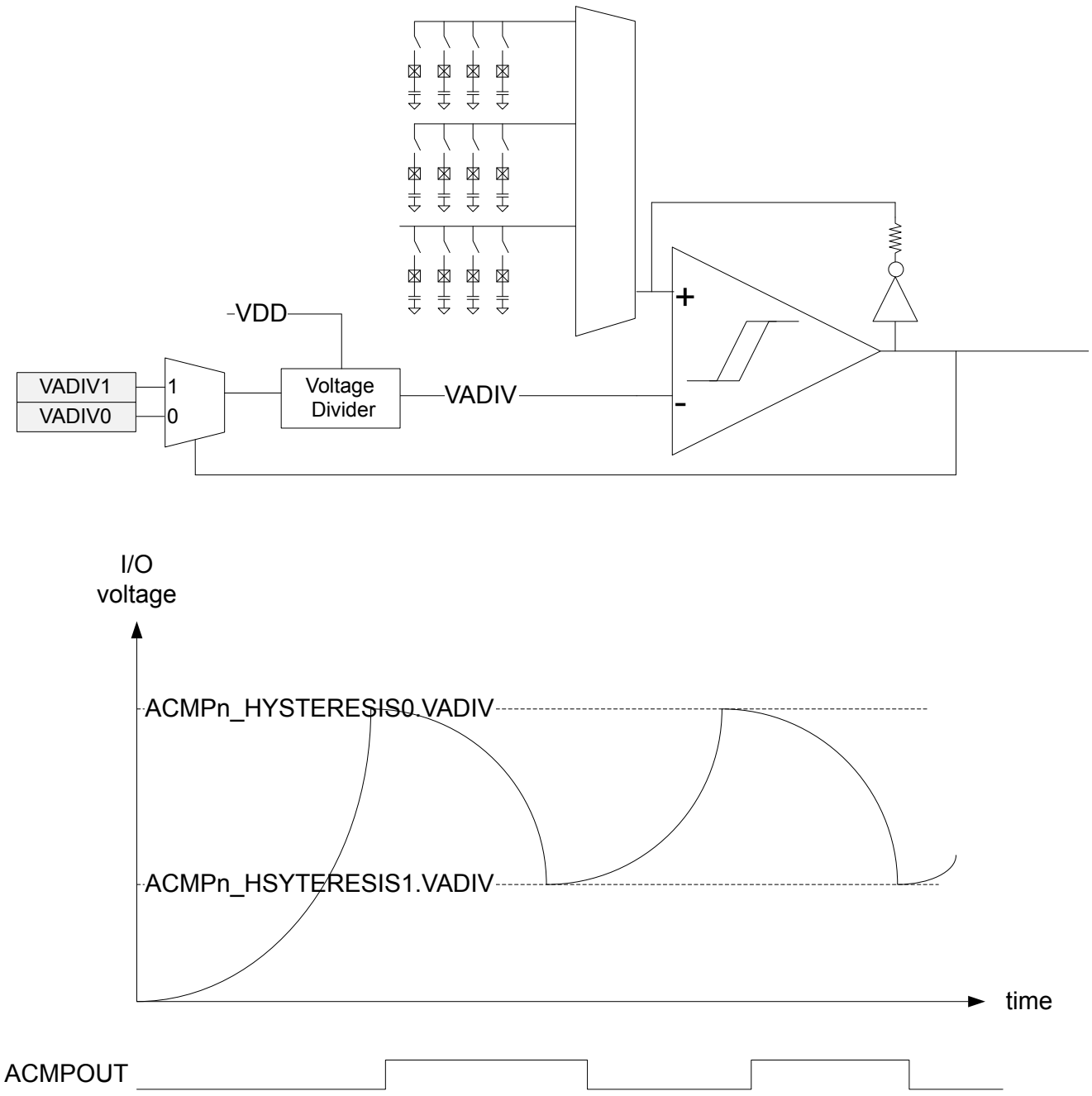


Figure 27.5. Capacitive Sensing Setup

27.3.8 Interrupts and PRS Output

The analog comparator includes an edge triggered interrupt flag (EDGE in ACMPn_IF). If either IRISE and/or IFALL in ACMPn_CTRL is set, the EDGE interrupt flag will be set on rising and/or falling edge of the comparator output respectively. An interrupt request will be sent if the EDGE interrupt flag in ACMPn_IF is set and enabled through the EDGE bit in ACMPn_IEN. The edge interrupt can also be used to wake up the device from EM3 Stop-EM1 Sleep.

The analog comparator includes the interrupt flag WARMUP in ACMPn_IF which is set when a warm-up sequence has finished. An interrupt request will be sent if the WARMUP interrupt flag in ACMPn_IF is set and enabled through the WARMUP bit in ACMPn_IEN.

The analog comparator can also generate an interrupt if a bus conflict occurs. An interrupt request will be sent if the APORTCONFLICT interrupt flag in ACMPn_IF is set and enabled through the APORTCONFLICT bit in ACMPn_IEN.

The synchronized comparator output is also available as a PRS output signal.

27.3.9 Output to GPIO

The output from the comparator and the capacitive sense output are available as alternate functions to the GPIO pins. Set the ACMPPEN bit in ACMPn_ROUTE to enable the output to a pin and the LOCATION bits to select the output location. The GPIO-pin must also be set as output. The output to the GPIO can be inverted by setting the GPIOINV bit in ACMPn_CTRL.

27.3.10 APORT Conflicts

The analog comparator connects to chip pins through APORT buses. It is possible that another APORT client is using a given APORT bus. To help debugging over-utilization of APORT resources the ACMP provides a number of status registers. The ACMPn_APORTREQ gives the user visibility into what APORT buses the ACMP is requesting given the setting of registers ACMPn_INPUTSEL and ACMPn_CTRL. ACMPn_APORTCONFLICT indicates if any of the selections are in conflict, internally or externally.

For example, if the user selects APORT1XCH0 for POSSEL and APORT3XCH1 for NEGSEL, then bits APORT1XCONFLICT and APORT3XCONFLICT would be 1 in register ACMPn_APORTCONFLICT, as it is illegal for POSSEL and NEGSEL to both select an X-bus simultaneously.

If the user wishes the ACMP to monitor the same pin as another APORT client within the system, the ACMP can be configured to not attempt to control the switches on an APORT bus via the fields APORTXMASTERDIS, APORTYMASTERDIS, and APORTVMASTERDIS in ACMPn_CTRL. APORTXMASTERDIS and APORTYMASTERDIS control if the X or Y bus selected via POSSEL or NEGESEL is mastered or not. APORTVMASTERDIS controls if either the X or Y bus selection of VASEL is mastered or not. When bus mastering is disabled, it is the other APORT client that determines which pin is connected to the APORT bus.

27.3.11 Supply Voltage Monitoring

The ACMP can be used to monitor supply voltages. The ACMP can select which voltage it uses via PWRSEL in ACMPn_CTRL. This voltage can be selected for VADIV using VASEL=0 in ACMPn_INPUTSEL and divided to a voltage with the band-gap reference range using DIVVA in registers ACMPn_HYSTERESIS0/1. The band-gap reference voltage can also be scaled via DIVVB in registers ACMPn_HYSTERESIS0/1 to provide a voltage higher or lower than the scaled VA voltage for comparison.

27.3.12 External Override Interface

The ACMP can be controlled by an external module, for instance LESENSE. In this mode, the external module will take control of the positive input mux control signal, which is normally controlled by ACMP_INPUTSEL_POSSEL. Only the APORTs are selectable for the positive input mux in this mode. Which APORT(s) used is configured in ACMP_EXTIFCTRL_APORTSEL. Additionally, the VLP should not be selected for the negative input mux in this mode.

Note: When the ACMP is controlled by the external interace, the ACMP warmup time may take up to 30 µs.

ACMP_EXTIFCTRL_APORTSEL also controls the base value for the positive input mux control signal. The external module will be able to add an offset to this base. The resulting mux configuration can be calculated using Figure 27.6 POSSEL in External Override Mode on page 989. The external module controls EXT_OFFSET, while EXT_BASE is controlled by ACMP. See register description of ACMP_EXTIFCTRL_APORTSEL to see values of EXT_BASE.

POSSEL = EXT_BASE + EXT_OFFSET

Figure 27.6. POSSEL in External Override Mode

Note: If only one APORT in a pair is used, the external module needs to be programmed to only use the channels that the ACMP has control of.

The external module is also able to override DIVVA and DIVVB in ACMP_HYSTERESIS0/HYSTERESIS1. This needs to be enabled in the external module. If the external module does not override DIVVA/DIVVB, the configuration in ACMP_HYSTERESIS0/HYSTERESIS1 will be used.

To enable the external override interface these steps must be performed:

- Configure the parts of the ACMP that will not be overridden, i.e. everything except ACMP_INPUTSEL_POSSEL and possibly ACMP_HYSTERESIS0/HYSTERESIS1. Make sure ACMP_CTRL_EN is set.
- Configure and enable the external override interface in ACMP_EXTIFCTRL.
- Check for APORT conflicts in ACMP_APORTCONFLICT.
- Wait for ACMP_STATUS_EXTIFACT to go high, indicating that the interface is ready to use.

27.4 Register Map

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

Offset	Name	Type	Description
0x000	ACMPn_CTRL	RW	Control Register
0x004	ACMPn_INPUTSEL	RW	Input Selection Register
0x008	ACMPn_STATUS	R	Status Register
0x00C	ACMPn_IF	R	Interrupt Flag Register
0x010	ACMPn_IFS	W1	Interrupt Flag Set Register
0x014	ACMPn_IFC	(R)W1	Interrupt Flag Clear Register
0x018	ACMPn_IEN	RW	Interrupt Enable Register
0x020	ACMPn_APORTREQ	R	APORT Request Status Register
0x024	ACMPn_APORTCONFLICT	R	APORT Conflict Status Register
0x028	ACMPn_HYSTERESIS0	RW	Hysteresis 0 Register
0x02C	ACMPn_HYSTERESIS1	RW	Hysteresis 1 Register
0x040	ACMPn_ROUTEPEN	RW	I/O Routing Pine Enable Register
0x044	ACMPn_ROUTELOC0	RW	I/O Routing Location Register
0x048	ACMPn_EXTIFCTRL	RW	External Override Interface Control

27.5 Register Description

27.5.1 ACMPn_CTRL - 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	0		0x07								0	0	0x0				0	0x0				0	0	0					0	0		0
Access	RW		RW								RW	RW	RW				RW	RW				RW	RW	RW					RW	RW		RW
Name	FULLBIAS		BIASPROG								IFALL	IRISE	INPUTRANGE				ACCURACY	PWRSEL				APORTVMASTERDIS	APORTYMASTERDIS	APORTXMASTERDIS					GPIOINV	INACTVAL		EN

Bit	Name	Reset	Access	Description
31	FULLBIAS	0	RW	**Full Bias Current**
	Set this bit to 1 for full bias current. See the data sheet for details.			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29:24	BIASPROG	0x07	RW	**Bias Configuration**
	These bits control the bias current level. See the data sheet for details.			
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	IFALL	0	RW	**Falling Edge Interrupt Sense**
	Set this bit to 1 to set the EDGE interrupt flag on falling edges of comparator output.			

Value	Mode	Description
0	DISABLED	Interrupt flag is not set on falling edges
1	ENABLED	Interrupt flag is set on falling edges

20	IRISE	0	RW	**Rising Edge Interrupt Sense**
	Set this bit to 1 to set the EDGE interrupt flag on rising edges of comparator output.			

Value	Mode	Description
0	DISABLED	Interrupt flag is not set on rising edges
1	ENABLED	Interrupt flag is set on rising edges

19:18	INPUTRANGE	0x0	RW	**Input Range**
	Adjust performance of the comparator for a given input voltage range.			

Value	Mode	Description
0	FULL	Setting when the input can be from 0 to ACMPVDD.
1	GTVDDDIV2	Setting when the input will always be greater than ACMPVDD/2.

Bit	Name	Reset	Access	Description
	2	LTVDDDIV2		Setting when the input will always be less than ACMPVDD/2.
17:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	ACCURACY	0	RW	**ACMP Accuracy Mode**

Select between low and high accuracy mode of the comparator. Note, high frequency changes can cause the ACMP performance to degrade. For such uses, such as quickly scanning through multiple channels or setting the ACMP to oscillate for capacitive sense, this bit should be set to 1.

Value	Mode	Description
0	LOW	ACMP operates in low-accuracy mode but consumes less current.
1	HIGH	ACMP operates in high-accuracy mode but consumes more current.

14:12	PWRSEL	0x0	RW	**Power Select**

Selects the power source for the ACMP(ACMPVDD). NOTE, this field should only be changed when the block is disabled (EN=0).

Value	Mode	Description
0	AVDD	AVDD supply
1	DVDD	DVDD supply
2	IOVDD0	IOVDD/IOVDD0 supply
4	IOVDD1	IOVDD1 supply (if part has two I/O voltages)

11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	APORTVMASTERDIS	0	RW	**APORT Bus Master Disable for Bus Selected By VASEL**

Determines if the ACMP will request the X or Y APORT bus selected by VASEL. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously by allowing the ACMP to not master the selected bus. When 1, the determination is expected to be from another peripheral, and the ACMP only passively looks at the bus. When 1, the selection of channel for a selected bus is ignored (the bus is not), and is whatever selection the external device mastering the bus has configured for the APORT bus.

Value	Description
0	Bus mastering enabled
1	Bus mastering disabled

9	APORTYMASTERDIS	0	RW	**APORT Bus Y Master Disable**

Determines if the ACMP will request the APORT Y bus selected by POSSEL or NEGSEL. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously by allowing the ACMP to not master the selected bus. When 1, the determination is expected to be from another peripheral, and the ACMP only passively looks at the bus. When 1, the selection of channel for a selected bus is ignored (the bus is not), and is whatever selection the external device mastering the bus has configured for the APORT bus.

Value	Description
0	Bus mastering enabled
1	Bus mastering disabled

Bit	Name	Reset	Access	Description
8	APORTXMASTER-DIS	0	RW	**APORT Bus X Master Disable**

Determines if the ACMP will request the APORT X bus selected by POSSEL or NEGSEL. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously by allowing the ACMP to not master the selected bus. When 1, the determination is expected to be from another peripheral, and the ACMP only passively looks at the bus. When 1, the selection of channel for a selected bus is ignored (the bus is not), and is whatever selection the external device mastering the bus has configured for the APORT bus.

Value	Description
0	Bus mastering enabled
1	Bus mastering disabled

7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	GPIOINV	0	RW	**Comparator GPIO Output Invert**

Set this bit to 1 to invert the comparator alternate function output to GPIO.

Value	Mode	Description
0	NOTINV	The comparator output to GPIO is not inverted
1	INV	The comparator output to GPIO is inverted

2	INACTVAL	0	RW	**Inactive Value**

The value of this bit is used as the comparator output when the comparator is inactive.

Value	Mode	Description
0	LOW	The inactive value is 0
1	HIGH	The inactive state is 1

1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**Analog Comparator Enable**

Enable/disable analog comparator.

27.5.2 ACMPn_INPUTSEL - Input Selection 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				0		0		0	0x00						0x00								0x00							
Access		RW				RW		RW		RW	RW						RW								RW							
Name		CSRESSEL				CSRESEN		VLPSEL		VBSEL	VASEL						NEGSEL								POSSEL							

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	CSRESSEL	0x0	RW	**Capacitive Sense Mode Internal Resistor Select**

These bits select the resistance value for the internal capacitive sense resistor. Resulting actual resistor values are given in the device data sheets.

Value	Mode	Description
0	RES0	Internal capacitive sense resistor value 0
1	RES1	Internal capacitive sense resistor value 1
2	RES2	Internal capacitive sense resistor value 2
3	RES3	Internal capacitive sense resistor value 3
4	RES4	Internal capacitive sense resistor value 4
5	RES5	Internal capacitive sense resistor value 5
6	RES6	Internal capacitive sense resistor value 6
7	RES7	Internal capacitive sense resistor value 7

27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	CSRESEN	0	RW	**Capacitive Sense Mode Internal Resistor Enable**

Enable/disable the internal capacitive sense resistor.

25	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
24	VLPSEL	0	RW	**Low-Power Sampled Voltage Selection**

Select the input to the sampled voltage VLP

Value	Mode	Description
0	VADIV	VADIV
1	VBDIV	VBDIV

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

Bit	Name	Reset	Access	Description
22	VBSEL	0	RW	**VB Selection**
	Select the input for the VB Divider			

Value	Mode	Description
0	1V25	1.25V
1	2V5	2.50V

Bit	Name	Reset	Access	Description
21:16	VASEL	0x00	RW	**VA Selection**
	Select the input for the VA Divider			

Mode	Value	Description
VDD	0x0	ACMPVDD
APORT2YCH0	0x1	APORT2Y Channel 0
APORT2YCH2	0x3	APORT2Y Channel 2
APORT2YCH4	0x5	APORT2Y Channel 4
. . .	. . .	. . .
APORT2YCH30	0x1f	APORT2Y Channel 30
APORT1XCH0	0x20	APORT1X Channel 0
APORT1YCH1	0x21	APORT1Y Channel 1
APORT1XCH2	0x22	APORT1X Channel 2
APORT1YCH3	0x23	APORT1Y Channel 3
APORT1XCH4	0x24	APORT1X Channel 4
APORT1YCH5	0x25	APORT1Y Channel 5
. . .	. . .	. . .
APORT1XCH30	0x3e	APORT1X Channel 30
APORT1YCH31	0x3f	APORT1Y Channel 31

Bit	Name	Reset	Access	Description
15:8	NEGSEL	0x00	RW	**Negative Input Select**
	Select negative input.			

APORT0XCH0	0x00	Dedicated APORT0X Channel 0
APORT0XCH1	0x01	Dedicated APORT0X Channel 1
APORT0XCH2	0x02	Dedicated APORT0X Channel 2
. . .	. . .	. . .
APORT0XCH15	0x0f	Dedicated APORT0X Channel 15
APORT0YCH0	0x10	Dedicated APORT0Y Channel 0
APORT0YCH1	0x11	Dedicated APORT0Y Channel 1
APORT0YCH2	0x12	Dedicated APORT0Y Channel 2
. . .	. . .	. . .
APORT0YCH15	0x1f	Dedicated APORT0Y Channel 15

Bit	Name	Reset	Access	Description
	APORT1XCH0	0x20		APORT1X Channel 0
	APORT1YCH1	0x21		APORT1Y Channel 1
	APORT1XCH2	0x22		APORT1X Channel 2
	APORT1YCH3	0x23		APORT1Y Channel 3
	APORT1XCH4	0x24		APORT1X Channel 4
	APORT1YCH5	0x25		APORT1Y Channel 5
	. . .	. . .		. . .
	APORT1XCH30	0x3e		APORT1X Channel 30
	APORT1YCH31	0x3f		APORT1Y Channel 31
	APORT2YCH0	0x40		APORT2Y Channel 0
	APORT2XCH1	0x41		APORT2X Channel 1
	APORT2YCH2	0x42		APORT2Y Channel 2
	APORT2XCH3	0x43		APORT2X Channel 3
	APORT2YCH4	0x44		APORT2Y Channel 4
	APORT2XCH5	0x45		APORT2X Channel 5
	. . .	. . .		. . .
	APORT2YCH30	0x5e		APORT2Y Channel 30
	APORT2XCH31	0x5f		APORT2X Channel 31
	APORT3XCH0	0x60		APORT3X Channel 0
	APORT3YCH1	0x61		APORT3Y Channel 1
	APORT3XCH2	0x62		APORT3X Channel 2
	APORT3YCH3	0x63		APORT3Y Channel 3
	APORT3XCH4	0x64		APORT3X Channel 4
	APORT3YCH5	0x65		APORT3Y Channel 5
	. . .	. . .		. . .
	APORT3XCH30	0x7e		APORT3X Channel 30
	APORT3YCH31	0x7f		APORT3Y Channel 31
	APORT4YCH0	0x80		APORT4Y Channel 0
	APORT4XCH1	0x81		APORT4X Channel 1
	APORT4YCH2	0x82		APORT4Y Channel 2
	APORT4XCH3	0x83		APORT4X Channel 3
	APORT4YCH4	0x84		APORT4Y Channel 4
	APORT4XCH5	0x85		APORT4X Channel 5
	. . .	. . .		. . .
	APORT4YCH30	0x9e		APORT4Y Channel 30
	APORT4XCH31	0x9f		APORT4X Channel 31
	DACOUT0	0xf2		DAC Channel 0 Output

Bit	Name	Reset	Access	Description
	DACOUT1	0xf3		DAC Channel 1 Output
	VLP	0xfb		Low-Power Sampled Voltage
	VBDIV	0xfc		Divided VB Voltage
	VADIV	0xfd		Divided VA Voltage
	VDD	0xfe		ACMPVDD as selected via PWRSEL
	VSS	0xff		VSS
7:0	POSSEL	0x00	RW	**Positive Input Select**
	Select positive input.			
	APORT0XCH0	0x00		Dedicated APORT0X Channel 0
	APORT0XCH1	0x01		Dedicated APORT0X Channel 1
	APORT0XCH2	0x02		Dedicated APORT0X Channel 2
	. . .	. . .		. . .
	APORT0XCH15	0x0f		Dedicated APORT0X Channel 15
	APORT0YCH0	0x10		Dedicated APORT0Y Channel 0
	APORT0YCH1	0x11		Dedicated APORT0Y Channel 1
	APORT0YCH2	0x12		Dedicated APORT0Y Channel 2
	. . .	. . .		. . .
	APORT0YCH15	0x1f		Dedicated APORT0Y Channel 15
	APORT1XCH0	0x20		APORT1X Channel 0
	APORT1YCH1	0x21		APORT1Y Channel 1
	APORT1XCH2	0x22		APORT1X Channel 2
	APORT1YCH3	0x23		APORT1Y Channel 3
	APORT1XCH4	0x24		APORT1X Channel 4
	APORT1YCH5	0x25		APORT1Y Channel 5
	. . .	. . .		. . .
	APORT1XCH30	0x3e		APORT1X Channel 30
	APORT1YCH31	0x3f		APORT1Y Channel 31
	APORT2YCH0	0x40		APORT2Y Channel 0
	APORT2XCH1	0x41		APORT2X Channel 1
	APORT2YCH2	0x42		APORT2Y Channel 2
	APORT2XCH3	0x43		APORT2X Channel 3
	APORT2YCH4	0x44		APORT2Y Channel 4
	APORT2XCH5	0x45		APORT2X Channel 5
	. . .	. . .		. . .
	APORT2YCH30	0x5e		APORT2Y Channel 30
	APORT2XCH31	0x5f		APORT2X Channel 31

Bit	Name	Reset	Access	Description
	APORT3XCH0	0x60		APORT3X Channel 0
	APORT3YCH1	0x61		APORT3Y Channel 1
	APORT3XCH2	0x62		APORT3X Channel 2
	APORT3YCH3	0x63		APORT3Y Channel 3
	APORT3XCH4	0x64		APORT3X Channel 4
	APORT3YCH5	0x65		APORT3Y Channel 5
	. . .	. . .		. . .
	APORT3XCH30	0x7e		APORT3X Channel 30
	APORT3YCH31	0x7f		APORT3Y Channel 31
	APORT4YCH0	0x80		APORT4Y Channel 0
	APORT4XCH1	0x81		APORT4X Channel 1
	APORT4YCH2	0x82		APORT4Y Channel 2
	APORT4XCH3	0x83		APORT4X Channel 3
	APORT4YCH4	0x84		APORT4Y Channel 4
	APORT4XCH5	0x85		APORT4X Channel 5
	. . .	. . .		. . .
	APORT4YCH30	0x9e		APORT4Y Channel 30
	APORT4XCH31	0x9f		APORT4X Channel 31
	DACOUT0	0xf2		DAC Channel 0 Output
	DACOUT1	0xf3		DAC Channel 1 Output
	R5VOUT	0xf4		R5V output
	VLP	0xfb		Low-Power Sampled Voltage
	VBDIV	0xfc		Divided VB Voltage
	VADIV	0xfd		Divided VA Voltage
	VDD	0xfe		ACMPVDD as selected via PWRSEL
	VSS	0xff		VSS

27.5.3 ACMPn_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																													0	0	0	0
Access																													R	R	R	R
Name																													EXTIFACT	APORTCONFLICT	ACMPOUT	ACMPACT

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	EXTIFACT	0	R	**External Override Interface Active**
	This bit is set when the external override interface is ready to use.			
2	APORTCONFLICT	0	R	**APORT Conflict Output**
	1 if any of the APORT BUSes being requested by the ACMPn are also being requested by another peripheral			
1	ACMPOUT	0	R	**Analog Comparator Output**
	Analog comparator output value.			
0	ACMPACT	0	R	**Analog Comparator Active**
	Analog comparator active status.			

27.5.4 ACMPn_IF - Interrupt Flag 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																														0	0	0
Access																														R	R	R
Name																														APORTCONFLICT	WARMUP	EDGE

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	APORTCONFLICT	0	R	**APORT Conflict Interrupt Flag**
	1 if any of the APORT BUSes being requested by the ACMPn are also being requested by another peripheral			
1	WARMUP	0	R	**Warm-up Interrupt Flag**
	Indicates that the analog comparator warm-up period is finished.			
0	EDGE	0	R	**Edge Triggered Interrupt Flag**
	Indicates that there has been a rising or falling edge on the analog comparator output.			

27.5.5 ACMPn_IFS - Interrupt Flag 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																														0	0	0
Access																														W1	W1	W1
Name																														APORTCONFLICT	WARMUP	EDGE

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	APORTCONFLICT	0	W1	**Set APORTCONFLICT Interrupt Flag**
	Write 1 to set the APORTCONFLICT interrupt flag			
1	WARMUP	0	W1	**Set WARMUP Interrupt Flag**
	Write 1 to set the WARMUP interrupt flag			
0	EDGE	0	W1	**Set EDGE Interrupt Flag**
	Write 1 to set the EDGE interrupt flag			

27.5.6 ACMPn_IFC - Interrupt Flag 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																														0	0	0
Access																														(R)W1	(R)W1	(R)W1
Name																														APORTCONFLICT	WARMUP	EDGE

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	APORTCONFLICT	0	(R)W1	**Clear APORTCONFLICT Interrupt Flag**
	Write 1 to clear the APORTCONFLICT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	WARMUP	0	(R)W1	**Clear WARMUP Interrupt Flag**
	Write 1 to clear the WARMUP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	EDGE	0	(R)W1	**Clear EDGE Interrupt Flag**
	Write 1 to clear the EDGE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

27.5.7 ACMPn_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																														0	0	0
Access																														RW	RW	RW
Name																														APORTCONFLICT	WARMUP	EDGE

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	APORTCONFLICT	0	RW	**APORTCONFLICT Interrupt Enable**
	Enable/disable the APORTCONFLICT interrupt			
1	WARMUP	0	RW	**WARMUP Interrupt Enable**
	Enable/disable the WARMUP interrupt			
0	EDGE	0	RW	**EDGE Interrupt Enable**
	Enable/disable the EDGE interrupt			

27.5.8 ACMPn_APORTREQ - APORT Request 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																							0	0	0	0	0	0	0	0	0	0
Access																							R	R	R	R	R	R	R	R	R	R
Name																							APORT4YREQ	APORT4XREQ	APORT3YREQ	APORT3XREQ	APORT2YREQ	APORT2XREQ	APORT1YREQ	APORT1XREQ	APORT0YREQ	APORT0XREQ

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	APORT4YREQ	0	R	**1 If the Bus Connected to APORT4Y is Requested**
	Reports if the bus connected to APORT4Y is being requested from the APORT			
8	APORT4XREQ	0	R	**1 If the Bus Connected to APORT4X is Requested**
	Reports if the bus connected to APORT4X is being requested from the APORT			
7	APORT3YREQ	0	R	**1 If the Bus Connected to APORT3Y is Requested**
	Reports if the bus connected to APORT3Y is being requested from the APORT			
6	APORT3XREQ	0	R	**1 If the Bus Connected to APORT3X is Requested**
	Reports if the bus connected to APORT3X is being requested from the APORT			
5	APORT2YREQ	0	R	**1 If the Bus Connected to APORT2Y is Requested**
	Reports if the bus connected to APORT2Y is being requested from the APORT			
4	APORT2XREQ	0	R	**1 If the Bus Connected to APORT2X is Requested**
	Reports if the bus connected to APORT2X is being requested from the APORT			
3	APORT1YREQ	0	R	**1 If the Bus Connected to APORT1X is Requested**
	Reports if the bus connected to APORT1X is being requested from the APORT			
2	APORT1XREQ	0	R	**1 If the Bus Connected to APORT2X is Requested**
	Reports if the bus connected to APORT2X is being requested from the APORT			
1	APORT0YREQ	0	R	**1 If the Bus Connected to APORT0Y is Requested**
	Reports if the bus connected to APORT0Y is being requested from the APORT			
0	APORT0XREQ	0	R	**1 If the Bus Connected to APORT0X is Requested**
	Reports if the bus connected to APORT0X is being requested from the APORT			

27.5.9 ACMPn_APORTCONFLICT - APORT Conflict Status 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																							0	0	0	0	0	0	0	0	0	0
Access																							R	R	R	R	R	R	R	R	R	R
Name																							APORT4YCONFLICT	APORT4XCONFLICT	APORT3YCONFLICT	APORT3XCONFLICT	APORT2YCONFLICT	APORT2XCONFLICT	APORT1YCONFLICT	APORT1XCONFLICT	APORT0YCONFLICT	APORT0XCONFLICT

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	APORT4YCONFLICT	0	R	**1 If the Bus Connected to APORT4Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT4Y is is also being requested by another peripheral
8	APORT4XCONFLICT	0	R	**1 If the Bus Connected to APORT4X is in Conflict With Another Peripheral** Reports if the bus connected to APORT4X is is also being requested by another peripheral
7	APORT3YCONFLICT	0	R	**1 If the Bus Connected to APORT3Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT3Y is is also being requested by another peripheral
6	APORT3XCONFLICT	0	R	**1 If the Bus Connected to APORT3X is in Conflict With Another Peripheral** Reports if the bus connected to APORT3X is is also being requested by another peripheral
5	APORT2YCONFLICT	0	R	**1 If the Bus Connected to APORT2Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT2Y is is also being requested by another peripheral
4	APORT2XCONFLICT	0	R	**1 If the Bus Connected to APORT2X is in Conflict With Another Peripheral** Reports if the bus connected to APORT2X is is also being requested by another peripheral
3	APORT1YCONFLICT	0	R	**1 If the Bus Connected to APORT1X is in Conflict With Another Peripheral** Reports if the bus connected to APORT1X is is also being requested by another peripheral
2	APORT1XCONFLICT	0	R	**1 If the Bus Connected to APORT1X is in Conflict With Another Peripheral** Reports if the bus connected to APORT1X is is also being requested by another peripheral
1	APORT0YCONFLICT	0	R	**1 If the Bus Connected to APORT0Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT0Y is is also being requested by another peripheral

Bit	Name	Reset	Access	Description
0	APORT0XCONFLICT	0	R	**1 If the Bus Connected to APORT0X is in Conflict With Another Peripheral** Reports if the bus connected to APORT0X is is also being requested by another peripheral

27.5.10 ACMPn_HYSTERESIS0 - Hysteresis 0 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			0x00								0x00																		0x0			
Access			RW								RW																		RW			
Name			DIVVB								DIVVA																		HYST			

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	DIVVB	0x00	RW	**Divider for VB Voltage When ACMPOUT=0**
	Divider to scale VB when ACMPOUT=0. VBDIV = VB * (DIVVB+1)/64.			
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:16	DIVVA	0x00	RW	**Divider for VA Voltage When ACMPOUT=0**
	Divider to scale VA when ACMPOUT=0. VADIV = VA * (DIVVA+1)/64.			
15:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	HYST	0x0	RW	**Hysteresis Select When ACMPOUT=0**
	Select hysteresis level when comparator output is 0. The hysteresis levels can vary, please see the electrical characteristics for the device for more information.			

Value	Mode	Description
0	HYST0	No hysteresis
1	HYST1	14 mV hysteresis
2	HYST2	25 mV hysteresis
3	HYST3	30 mV hysteresis
4	HYST4	35 mV hysteresis
5	HYST5	39 mV hysteresis
6	HYST6	42 mV hysteresis
7	HYST7	45 mV hysteresis
8	HYST8	No hysteresis
9	HYST9	-14 mV hysteresis
10	HYST10	-25 mV hysteresis
11	HYST11	-30 mV hysteresis
12	HYST12	-35 mV hysteresis
13	HYST13	-39 mV hysteresis
14	HYST14	-42 mV hysteresis
15	HYST15	-45 mV hysteresis

27.5.11 ACMPn_HYSTERESIS1 - Hysteresis 1 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			0x00								0x00																		0x0			
Access			RW								RW																		RW			
Name			DIVVB								DIVVA																		HYST			

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	DIVVB	0x00	RW	**Divider for VB Voltage When ACMPOUT=1**
	Divider to scale VB when ACMPOUT=1. VBDIV = VB * (DIVVB+1)/64.			
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:16	DIVVA	0x00	RW	**Divider for VA Voltage When ACMPOUT=1**
	Divider to scale VA when ACMPOUT=1. VADIV = VA * (DIVVA+1)/64.			
15:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	HYST	0x0	RW	**Hysteresis Select When ACMPOUT=1**
	Select hysteresis level when comparator output is 1. The hysteresis levels can vary, please see the electrical characteristics for the device for more information.			

Value	Mode	Description
0	HYST0	No hysteresis
1	HYST1	14 mV hysteresis
2	HYST2	25 mV hysteresis
3	HYST3	30 mV hysteresis
4	HYST4	35 mV hysteresis
5	HYST5	39 mV hysteresis
6	HYST6	42 mV hysteresis
7	HYST7	45 mV hysteresis
8	HYST8	No hysteresis
9	HYST9	-14 mV hysteresis
10	HYST10	-25 mV hysteresis
11	HYST11	-30 mV hysteresis
12	HYST12	-35 mV hysteresis
13	HYST13	-39 mV hysteresis
14	HYST14	-42 mV hysteresis
15	HYST15	-45 mV hysteresis

27.5.12 ACMPn_ROUTEPEN - I/O Routing Pine 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																																0
Access																																RW
Name																																OUTPEN

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	OUTPEN	0	RW	**ACMP Output Pin Enable**
	Enable/disable analog comparator output to pin.			

27.5.13 ACMPn_ROUTELOC0 - I/O Routing Location 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																											0x00					
Access																											RW					
Name																											OUTLOC					

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	OUTLOC	0x00	RW	**I/O Location**
	Decides the location of the OUT pin.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6
7	LOC7	Location 7

27.5.14 ACMPn_EXTIFCTRL - External Override Interface Control

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							0
Access																									RW							RW
Name																									APORTSEL							EN

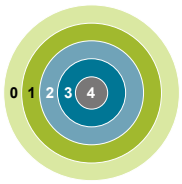
Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:4	APORTSEL	0x0	RW	**APORT Selection for External Interface**

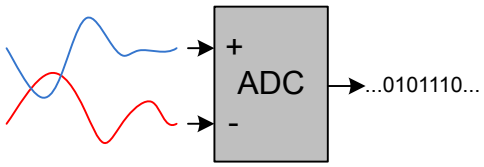
Decides which APORT(s) the ACMP will use when controlled by an external module.

Value	Mode	Description
0	APORT0X	APORT0X used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT0XCH0.
1	APORT0Y	APORT0Y used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT0YCH0.
2	APORT1X	APORT1X used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT1XCH0.
3	APORT1Y	APORT1Y used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT1XCH0.
4	APORT1XY	APORT1X/Y used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT1XCH0.
5	APORT2X	APORT2X used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT2YCH0.
6	APORT2Y	APORT2Y used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT2YCH0.
7	APORT2YX	APORT2Y/X used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT2YCH0.
8	APORT3X	APORT3X used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT3XCH0.
9	APORT3Y	APORT3Y used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT3XCH0.
10	APORT3XY	APORT3X/Y used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT3XCH0.
11	APORT4X	APORT4X used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT4YCH0.
12	APORT4Y	APORT4Y used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT4YCH0.
13	APORT4YX	APORT4Y/X used. EXT_BASE = ACMP_INPUTSEL_POSSEL_APORT4YCH0.

Bit	Name	Reset	Access	Description
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**Enable External Interface**
	Set to enable an external module, like LESENSE, to control the ACMP			

28. ADC - Analog to Digital Converter





Quick Facts

What?

The ADC is used to convert analog signals into a digital representation and features low-power, autonomous 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?

A low power ADC samples up to 32 input channels in a programmable sequence. With the help of PRS and DMA, the ADC can operate without CPU intervention in EM2 DeepSleep and EM3 Stop, minimizing the number of powered up resources. The ADC can further be duty-cycled to reduce the energy consumption.

28.1 Introduction

The ADC uses a Successive Approximation Register (SAR) architecture, with a resolution of up to 12 bits at up to one million samples per second (1 Msps). The integrated input multiplexer can select from external I/Os and several internal signals.

28.2 Features

- Programmable resolution (6/8/12-bit)
  - 13 conversion clock cycles for a 12-bit conversion
  - Maximum 1 Msps @ 12-bit
  - Maximum 1.6 Msps @ 6-bit
- Configurable acquisition time
- Externally controllable conversion start time using PRS in TIMED mode
- Integrated prescaler for conversion clock generation
  - Selectable clock division factor from 1 to 128
- Wide conversion clock range: 32 kHz to 16 MHz
- Can be run during EM2 DeepSleep and EM3 Stop, waking up the system upon various enabled interrupts
- Can be run during EM2 DeepSleep and EM3 Stop with DMA enabled to pull data from the FIFOs without waking up the system
- Supports up to 144 external input channels and several internal inputs
  - Includes temperature sensor and random number generator function
- Left or right adjusted results
  - Results in 2's complement representation
  - Differential results sign extended to 32-bits results
- Programmable scan sequence
  - Up to 32 configurable samples in scan sequence
  - Mask to select which pins are included in the sequence
  - Triggered by software or PRS input
  - One shot or repetitive mode
  - Oversampling available
  - Four deep FIFO to store conversion data along with channel ID and option to overwrite old data when full
  - Programmable watermark (DVL) to generate SCAN interrupt
  - Supports overflow and underflow interrupt generation
  - Supports window compare function
  - Conversion tailgating support for predictable periodic scans
- Programmable single channel conversion
  - Triggered by software or PRS input
  - Can be interleaved between two scan sequences
  - One shot or repetitive mode
  - Oversampling available
  - Four deep FIFO to store conversion data with option to overwrite old data when full
  - programmable watermark (DVL) to generate SINGLE interrupt
  - Supports overflow and underflow interrupt generation
  - Supports window compare function
- Hardware oversampling support
  - 1st order accumulate and dump filter
  - From 2 to 4096 oversampling ratio (OSR)
  - Results in 16-bit representation
  - Enabled individually for scan sequence and single channel mode
  - Common OSR select
- Programmable and preset input full scale (peak-to-peak) range (VFS) with selectable reference sources
  - VFS=1.25 V using internal VBGR reference
  - VFS=2.5 V using internal VBGR reference
  - VFS=AVDD with AVDD as reference source
  - VFS=5 V with internal VBGR reference
  - Single ended external reference
  - Differential external reference
  - VFS=2xAVDD with AVDD as reference source
  - User-programmable dividers for flexible VFS options from internal, external or supply voltage reference sources

- Support for offset and gain calibration
- Interrupt generation and/or DMA request when
  - Programmable number of converted data available in the single FIFO (also generates DMA request)
  - Programmable number of converted data available in the scan FIFO (also generates DMA request)
  - Single FIFO overflow or underflow
  - Scan FIFO overflow or underflow
  - Latest Single conversion tripped compare logic
  - Latest Scan conversion tripped compare logic
  - Analog over-voltage interrupt
  - Programming Error interrupt due to APORT Bus Request conflict or NEGSEL programming error

28.3 Functional Description

An overview of the ADC is shown in Figure 28.1 ADC Overview on page 1012.

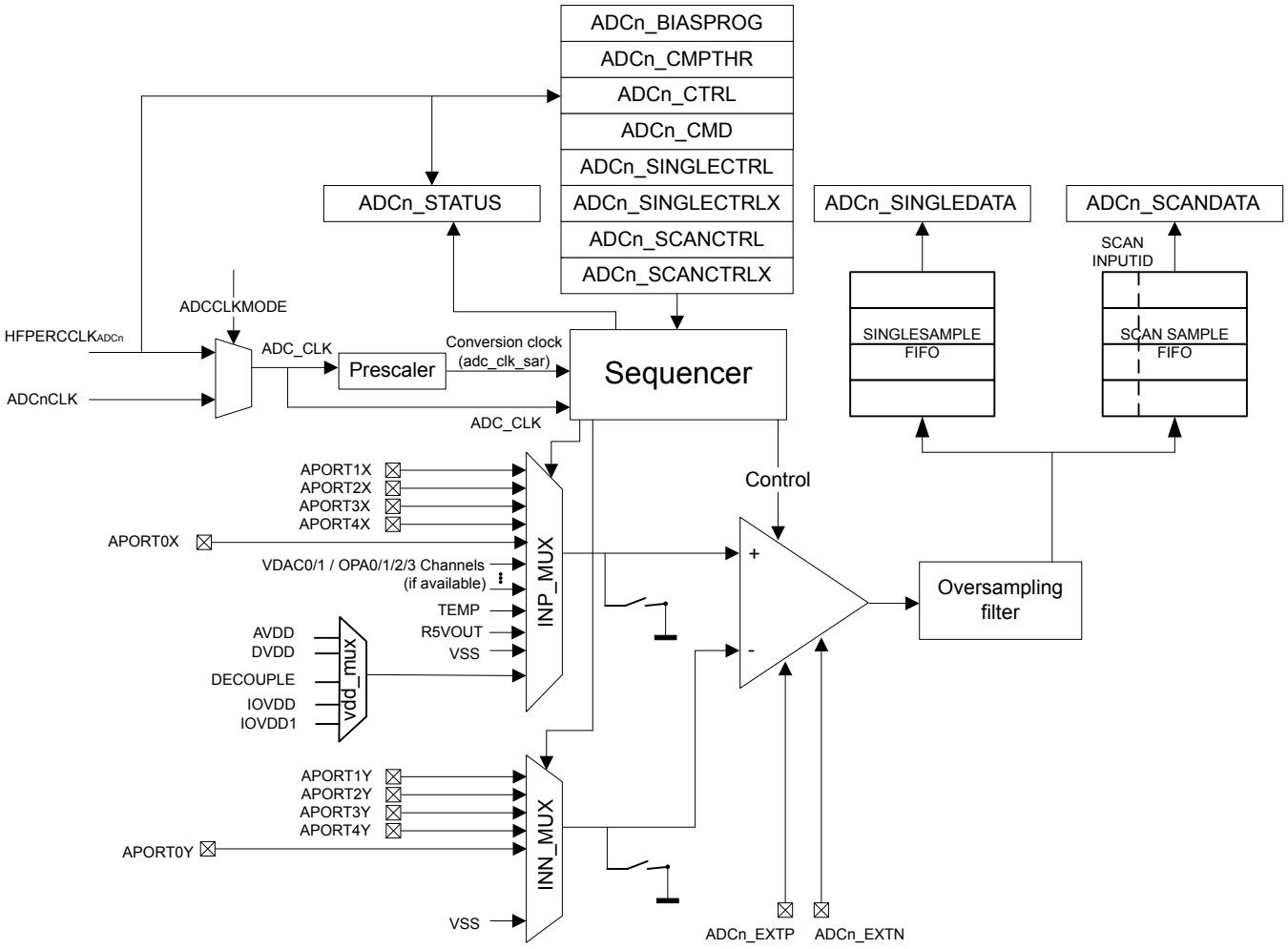


Figure 28.1. ADC Overview

28.3.1 Clock Selection

The ADC logic is partitioned into two clock domains: HFPERCCLK and ADC_CLK. The HFPERCCLK domain contains the register interface logic, APORT request logic and portions of FIFO read logic. The HFPERCCLK is the default clock for the ADC peripheral. The rest of the ADC is clocked by the ADC_CLK domain. The ADC_CLK is chosen by ADCCLKMODE bit in the ADCn_CTRL register.

The ADC_CLK is the main clock for the ADC engine. If the ADCCLKMODE is set to SYNC, the ADC_CLK is equal to the HFPERCCLK and the ADC operates in synchronous mode. If the ADCCLKMODE is set to ASYNC, the ADC_CLK is ASYNCCLK and the ADC operates in asynchronous mode. This distinction is important to understand as there are additional system restrictions and benefits to running the ADC in asynchronous mode detailed in 28.3.15 ASYNC ADC_CLK Usage Restrictions and Benefits.

Note: Whenever ADC is being used in asynchronous mode, then HFPERCLK must be at least 1.5 times higher than the ADC_CLK.

The ADC has an internal clock prescaler, controlled by PRESC bits in ADCn_CTRL, which can divide the ADC_CLK by any factor between 1 and 128 to generate the conversion clock (adc_clk_sar) for the ADC. This adc_clk_sar is also used to generate acquisition timing. Note that the maximum clock frequency for adc_clk_sar is 16 MHz. The ADC warmup time is determined by ADC_CLK and not by adc_clk_sar.

ASYNCCLK is a clock source from the CMU which is considered asynchronous to HFPERCCLK. The CMU_ADCCTRL register can be programmed to request and use ASYNCCLK. It has multiple choices for its source, including AUXHFRCO, HFXO and HFSRCCLK, and can optionally be inverted. If the chosen source for ASYNCCLK is not active at the time of request, the CMU enables the source oscillator upon receiving the request, and shuts down the oscillator when the ADC stops requesting the clock. Consult the CMU chapter for details of how to program the clock sources for the ASYNCCLK and oscillator start-up time details.

Software may choose a clock request generation scheme by programming the ASYNCCLKEN and WARMMODE of the ADCn_CTRL register. If the ASYNCCLKEN is set to ASNEEDED with WARMMODE set to NORMAL, the ADC requests ASYNCCLK only when a conversion trigger is activated. The ASYNCCLK request is withdrawn after the conversion is complete. All other options keep the ASYNCCLK request "ON" until software programs these fields otherwise or changes the ADCCLKMODE to SYNC.

For EM2 DeepSleep or EM3 Stop operation of the ADC, the ADC_CLK must be configured for AUXHFRCO as this is the only available option during EM2 DeepSleep or EM3 Stop. The ADC_CLK source should not be changed as the system enters or exits various energy modes, otherwise measurement inaccuracies will result.

28.3.2 Conversions

A conversion consists of two phases: acquisition and approximation. The input is sampled in the acquisition phase before it is converted to digital representation during the approximation phase. The acquisition time can be configured independently for scan sequence and single channel conversions (see 28.3.3 ADC Modes) by setting AT in ADCn_SINGLECTRL/ADCn_SCANCTRL. The acquisition times can be set to 1 , 2, 3 or any integer power of 2 from 4 to 256 adc_clk_sar cycles.

Note: For high impedance sources the acquisition time should be adjusted to allow enough time for the internal sample capacitor to fully charge. The minimum acquisition time for sampling at 1 Msps and typical input loading is 187.5 ns.

The ADC uses one adc_clk_sar cycle per output bit in the approximation phase plus 1 extra adc_clk_sar cycle.

$$T_{conv} = (T_{acq} + (N + 1) \times T_{adc\_clk\_sar}) \times OVSRSEL$$

Where T_{acq} is the acquisition time set by the AT bit field, N is the resolution (in bits), and OVSRSEL is the oversampling ratio according to the OVSRSEL field in ADCn_CTRL when oversampling is enabled (see 28.3.10.6 Oversampling).

Figure 28.2. ADC Total Conversion Time Per Output

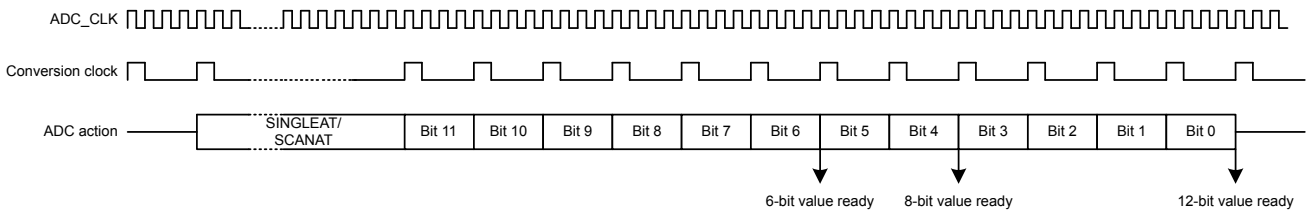


Figure 28.3. ADC Conversion Timing

28.3.3 ADC Modes

The ADC contains two programmable modes: single channel mode and scan mode. Both modes have separate configuration registers and a four-deep FIFO for conversion results. Both modes may be set up to run only once per trigger or to automatically repeat after each operation. The scan mode has priority over the single channel mode. However by default, if scan sequence is running, a triggered single channel conversion will be interleaved between two scan samples.

28.3.3.1 Single Channel Mode

Single channel mode can be used to convert a single channel either once per trigger or repetitively. The configuration of single channel mode is done using the ADCn_SINGLECTRL and ADCn_SINGLECTRLX registers and the result FIFO can be read through the ADCn_SINGLEDATA register. The DVL field of the ADCn_SINGLECTRLX controls the FIFO watermark crossing which sets the SINGLEDV bit in ADCn_STATUS high and is cleared when the data is read and the number of unread data samples falls below the DVL threshold. The user can choose to throw out new samples or overwrite the old samples when the FIFO becomes full by programming the FIFOOFACT field of the ADCn_SINGLECTRLX register. Single channel results can also be read through ADCn_SINGLEDATAP without popping the FIFO, returning its latest element. The DIFF field in ADCn_SINGLECTRL selects whether differential or single ended inputs are used and POSSEL and NEGSEL selects the input signal(s). The CMPEN bit in the ADCn_SINGLECTRL register enables the window compare function, and the latest converted data is compared against values programmed into the ADGT and ADLT fields of the ADCn_CMPTHR register and generates SINGLECMP interrupts if enabled. The window compare function allows for compare triggering both within (if ADGT less than ADLT) or out of (if ADGT greater than ADLT) window.

28.3.3.2 Scan Mode

Scan mode is used to perform conversions across multiple channels, sweeping a set of selected inputs in a sequence. The configuration of scan mode is done in the ADCn_SCANCTRL and ADCn_SCANCTRLX registers. It has similar controls and data read mechanisms to single channel mode. There are two key differences between single channel mode and scan mode: the input sequence is programmed differently, and it has additional information in the result to indicate the channel on which the conversion was acquired. 28.3.7 Input Selection explains how the input sequence is chosen. When the scan sequence is triggered, the ADC samples all inputs that are included in the mask (ADCn_SCANMASK), starting at the lowest pin number. DIFF in ADCn_SCANCTRL selects whether single ended or differential inputs are used. The FIFO data is tagged with SCANINPUTID and can be read along with the scan data using ADCn_SCANDATAX register. The ADCn_SCANDATAXP can be used to read the latest valid entry from the scan FIFO without popping it. There is also a ADCn_SCANDATA register that contains results without the SCANINPUTID appended.

Note: If using scan mode with ADCn_SCANCTRL_REP = 1 and ADCn_SCANCTRLX_REPDELAY = NODELAY, the last channel in the scan will report a SCANINPUTID of 0. Using ADCn_SCANCTRLX_REPDELAY with any value other than NODELAY will report the correct SCANINPUTID.

28.3.4 Warm-up Time

After power-on, the ADC requires some time for internal bias currents and references to settle prior to starting a conversion. This time period is called the warm-up time. Warm-up timing is performed by hardware. Software must program the number of ADC_CLK cycles required to count at least 1 µs in the TIMEBASE field of the ADCn_CTRL register. TIMEBASE only affects the timing of the warm-up sequence and is not dependent on adc_clk_sar. When enabling the ADC or changing references between samples, the ADC is automatically warmed up for 5 µs (5 times the period indicated by TIMEBASE).

Normally, the ADC will be warmed up only when samples are requested and is shut off when there are no more samples waiting. However, if lower latency is needed, configuring the WARMUPMODE field in ADCn_CTRL allows the ADC and/or reference to stay warm between samples, reducing the warm-up time or eliminating it altogether. Figure 28.4 ADC Analog Power Consumption With Different WARMUPMODE Settings on page 1016 shows the effects on analog power consumption in scenarios using different WARMUPMODE settings.

The user can program which reference should be kept warm in the CHCONREFWARMIDLE bitfield in the ADCn_CTRL register. By default the scan mode reference is kept warm. The user can also choose to keep the single channel mode reference warm or to keep the last used reference warm. If the default setting is kept (scan mode reference is to be kept warm) and if the single-mode reference setting is different than scan-mode, then single mode conversions will first warmup its reference for 5 µs before a conversion can begin.

Various warmup modes are described here:

- NORMAL: This is the lowest power option for general-purpose use and low sampling rates (below 35 ksps). The ADC and references are shut off when there are no samples waiting. The ADC does not consume any power when it is shut down. A 5 µs warmup time will be initiated prior to every conversion. Figure a in Figure 28.4 ADC Analog Power Consumption With Different WARMUPMODE Settings on page 1016 shows this mode.
- KEEPINSTANDBY: This mode is suitable for infrequent sampling of lower impedance inputs, and is the lowest power option for sampling rates between about 35 and 125 ksps. It may also be useful for lower sampling rates where latency is important. The reference selected for scan mode is kept warm, but the ADC is powered down. The ADC will initiate a 1 µs warmup period before a conversion begins. Because the reference is kept warm, the ADC will consume a small amount of standby current when it is not converting. Figure b in Figure 28.4 ADC Analog Power Consumption With Different WARMUPMODE Settings on page 1016 shows this mode.
- KEEPINSLOWACC: This mode is useful for high-impedance inputs which are sampled infrequently. It is similar to KEEPINSTANDBY, but continuously tracks the input, keeping the input multiplexer connected to the APORT bus. This mode consumes little more power than KEEPINSTANDBY mode (about 2 µA extra) when a conversion is not in progress. This allows the user to avoid programming long acquisition time that would otherwise be necessary for high-impedance inputs when ADC wakes up to full power mode, thereby reducing the total current consumption per conversion.
- KEEPADCWARM: This mode provides the lowest latency and allows for maximum sampling rates. The ADC and reference circuitry remain powered on even when conversions are not in progress. Figure c in Figure 28.4 ADC Analog Power Consumption With Different WARMUPMODE Settings on page 1016 shows this mode. This mode consumes the most power, but as soon as a trigger event occurs, the acquisition and conversion begin with no warm-up time. Note that if KEEPADCWARM mode is set and HFXO is selected as the ADC clock source, the HFXO will remain on in EM2.

When KEEPADCWARM is chosen, ADC is termed as being in continuous operation. When any other warmup mode is chosen, ADC is termed to be in duty-cycled operation.

When entering EM2 DeepSleep or EM3 Stop, if the ADC is not going to be used, it should be returned to an idle state and WARMUPMODE in ADCn_CTRL written to 0. Refer to 28.3.17 ADC Programming Model for more information on placing the ADC in an idle state. If the ADC is going to be used in these low energy modes, the user can use any of the WARMUPMODE settings, but should be mindful of the power consumption that comes along with the different mode settings. For EM2 DeepSleep or EM3 Stop operation, the ADC clock source must be configured to use AUXHFRCO.

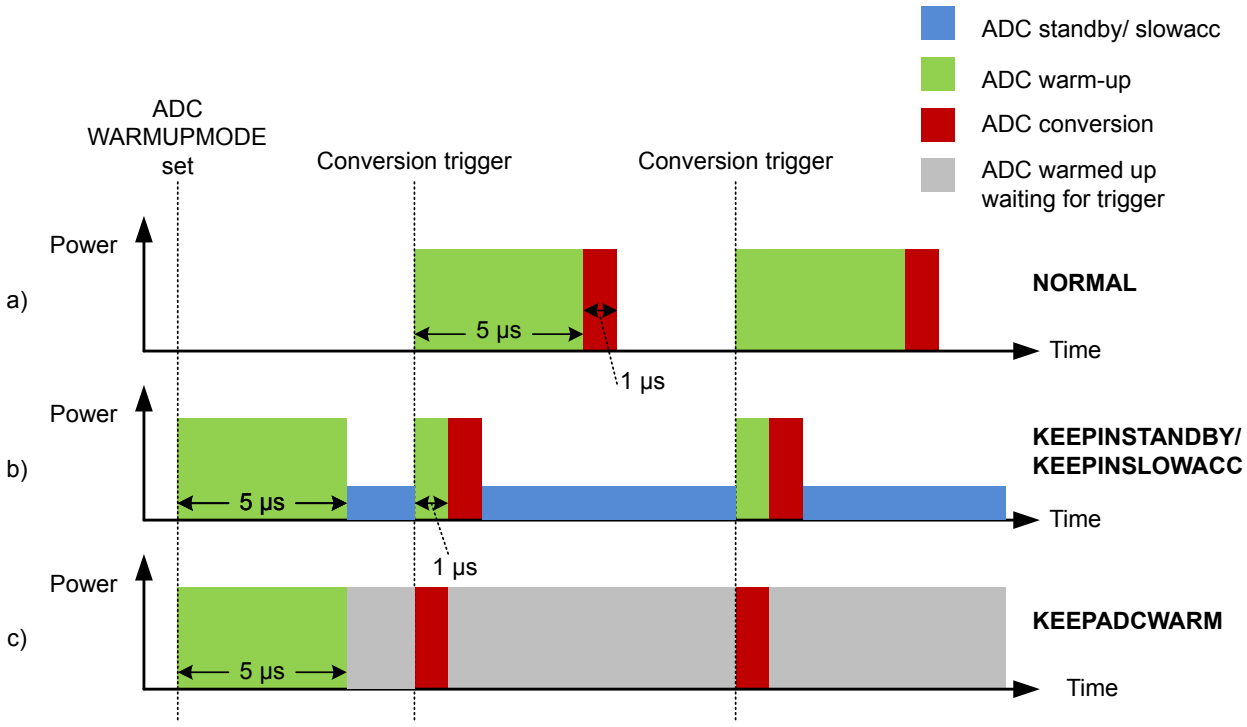


Figure 28.4. ADC Analog Power Consumption With Different WARMUPMODE Settings

Note: When using any warm-up mode other than NORMAL, always switch back to the NORMAL mode before switching to another warm-up mode.

28.3.5 Power Supply

The ADC block power (V_{ADC}) is derived from the VDDX_ANA supply rail. VDDX_ANA can be selected from the AVDD or DVDD supply pins using the EMU_PWRCTRL_ANASW bit field.

28.3.6 Input Pin Considerations

For external ADC inputs routed through the APORT, the maximum supported analog input voltage will be limited to the MIN(V_{ADC}, IOVDD) (where V_{ADC} is VDDX_ANA, as described in 28.3.5 Power Supply). Note that pins configured as ADC inputs should disable OVT (by setting the corresponding GPIO_Px_OVTDIS bit) to reduce any potential distortion introduced by the OVT circuitry.

ADC external reference inputs are not routed through the APORT, and the maximum supported analog input voltage for an external reference will also be limited to the MIN(V_{ADC}, IOVDD).

28.3.7 Input Selection

The ADC samples and converts the analog voltage differential at its positive and negative voltage inputs. The input multiplexers of the ADC can connect these inputs to one of several internal nodes (e.g., temperature sensor) or to external signals via analog ports (APORT0, APORT1, APORT2, APORT3 or APORT4).

The analog ports APORT1, APORT2, APORT3, and APORT4 connect to external pins via analog buses (BUSAX, BUSAY, BUSBX, etc.) which are shared among other analog peripherals on the device. APORT1 through APORT4 are each 32 channels wide with connections to two sub-buses: a 16-channel X bus and a 16-channel Y bus. In the ADC module, all X buses connect to the INP_MUX and all Y buses connect to the INN_MUX as shown in Figure 28.5 APORT Connection to the ADC on page 1017. Connections to the X and Y sub-buses alternate channels on the APORT. On APORT1 and APORT3, even-numbered channels connect to the X bus, and odd-numbered channels connect to the Y bus. On APORT2 and APORT4, even-numbered channels connect to the Y bus and odd-numbered channels connect to the X bus. The APORT to BUS mappings may vary from device to device, Refer to the APORT Client Map in the device data sheet for exact mappings.

Unlike APORT1 through APORT4, APORT0 is not a shared resource. It consists of a 16-channel X bus and a 16-channel Y bus, each with dedicated I/O pin connections. Note that APORT0 is not available on all device families.

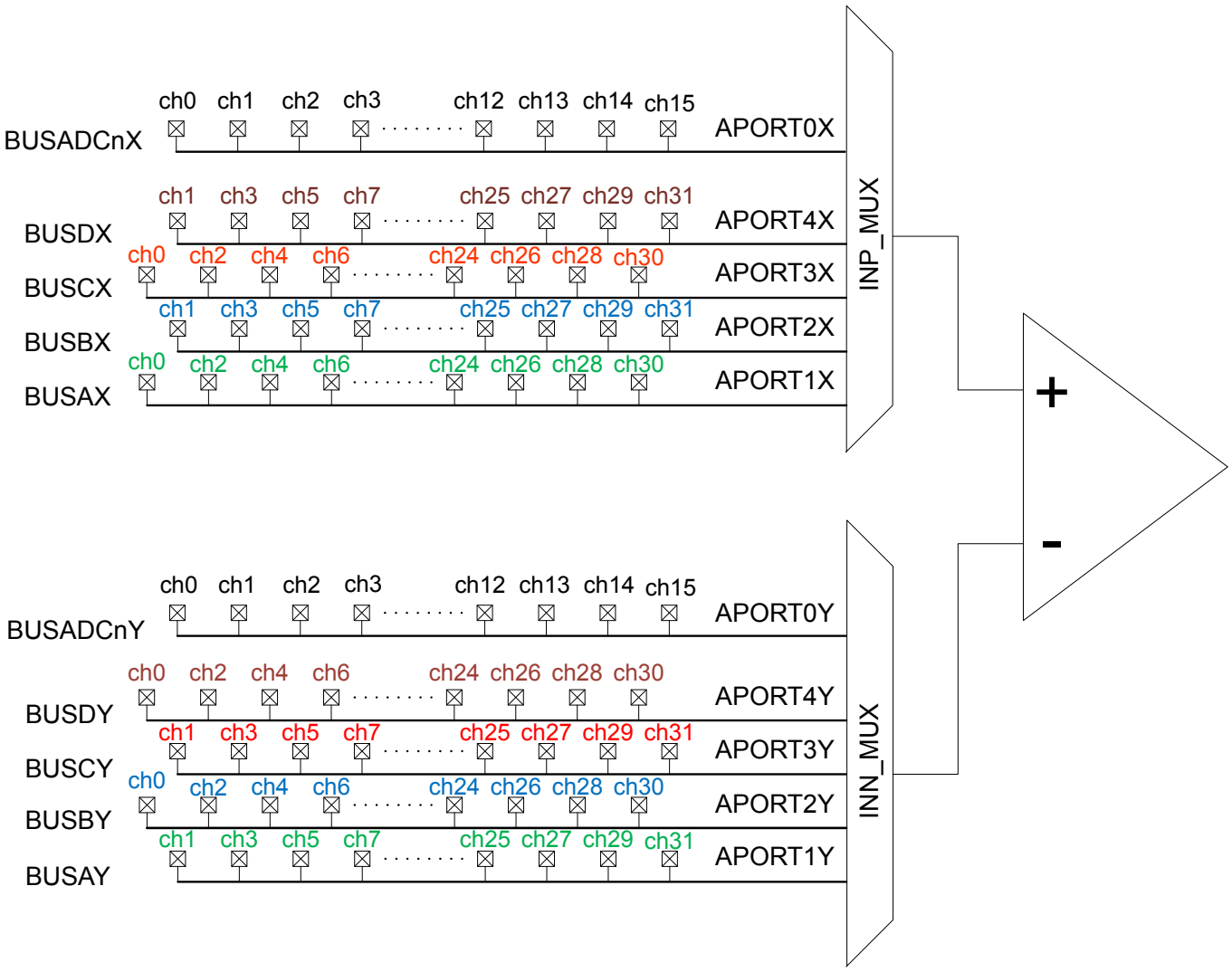


Figure 28.5. APORT Connection to the ADC

For differential measurements, one input must be chosen from an X bus and the other from a Y bus. Choosing both inputs from an X bus or both from a Y bus will generate a PROGERR interrupt (if enabled) of NEGSELCONF type. The PROGERR type can be checked in the ADCn_STATUS register.

The mapping and availability for external I/O connections to ADC0 inputs is shown in device data sheet.

Multiple peripherals may request the same shared system bus (BUSAX, BUSAY, BUSBX, etc.). When this happens, a conflict status is generated and that bus is kept floating. If this happens with the ADC, the PROGERR field in ADCn_STATUS is set to BUSCONF, and an interrupt may be generated (if enabled). When connecting dedicated I/Os through APORT0, all inputs are available to APORT0X and APORT0Y and no bus conflict is possible. Refer to 28.3.7.3 APORT Conflicts for more information on identifying and resolving bus conflicts.

Note: The internal inputs can only be sampled in single channel, single-ended mode. NEGSEL should be fixed to VSS for these conversions.

28.3.7.1 Configuring ADC Inputs in Single Channel Mode

In single channel mode, the ADCn_SINGLECTRL register provides the POSSEL and NEGSEL selection for positive and negative channel selection of the ADC. The APORT Client Map provides external pin to internal bus channel mapping enumeration for a particular device. Software can also choose internal nodes for POSSEL.

For single-ended conversions on external (APORT-connected) signals, POSSEL and NEGSEL are fully configurable. However, when performing conversions on internal signals, NEGSEL must be set to VSS. This NEGSEL reconfigurability feature in single-ended mode may not be available in all devices. If compatibility with devices that do not support this feature is desired, NEGSEL should be set to VSS for all single channel single-ended conversions.

Note that in both the POSSEL and NEGSEL fields, it is possible to choose inputs from both X and Y buses, even though X channels are physically connected to the positive mux (INP_MUX) and Y channels are physically connected to the negative mux (INN_MUX). For single-ended operation (DIFF = 0), if the positive input is chosen from a Y channel the ADC performs a negative single ended conversion and automatically inverts the result at the end, producing a positive result. For differential conversions (DIFF = 1), if a Y channel is chosen for the positive input and an X channel is chosen for the negative input, the ADC result will be inverted to produce the correct polarity.

Refer to device-specific data sheet for specific pin connection options. Note that the same I/O pin may appear in multiple locations.

28.3.7.2 Configuring ADC Inputs in Scan Mode

In scan mode, the ADC can sample and convert up to 32 external channels on each conversion trigger. Internal channels are not available in scan mode. The ADC's scanner logic automatically changes the input mux settings between conversions, eliminating the need for firmware intervention.

The ADC scanner logic is controlled by a set of 32 logical channels called SCANINPUTIDs. The 32 SCANINPUTIDs are arranged in four groups of 8 channels each. Each channel group can point to a predefined series of 8 sequential channels on any of the available APORTs. The ADCn_SCANINPUTSEL register is used to configure which group of physical APORT channels each of the SCANINPUTID channel groups map to. For example, selecting APORT1CH16TOCH23 in the INPUT7TO0SEL field selects APORT1CH16 for SCANINPUTID0, APORT1CH17 for SCANINPUTID1, APORT1CH18 for SCANINPUTID2, and so on.

The four SCANINPUTID groups are fully independent and may be selected from any APORT in any combination. It is possible also to repeat the same selection in multiple groups. For example, the user may select APORT2CH0TOCH7 for all four of the SCANINPUTID groups.

In many cases, the user application will not require all 32 channels of the scanner to be converted. Each of the scanner channels may be individually enabled according to the needs of the system. The ADCn_SCANMASK register is used to enable and disable individual SCANINPUTIDs. The bits in the ADCnSCANMASK register correspond one-to-one with the SCANINPUTID channel numbers. During a scan operation, the ADC scanner logic will convert only the enabled SCANINPUTIDs, in order from lowest to highest.

In single-ended mode, all conversions performed by the ADC will be relative to VSS. For any enabled SCANINPUTID, the selected APORT channel will be connected to the ADC with the opposite ADC input terminal connected to VSS. Note that the channel groups selected in ADCn_SCANINPUTSEL point to a block of 8 channels on an APORT, which includes both X and Y channels. Depending on the channels enabled by ADCn_SCANMASK, the ADC may perform conversions on the X or the Y bus associated with that APORT.

Figure 28.6 ADC Single-ended Scan Mode Example on page 1019 shows an example of a single-ended scan configuration. In this example, ADCn_SCANINPUTSEL has been configured to place APORT1CH16TO23 in the first, third, and fourth channel groups. APORT4CH8TO15 has been placed in the second channel group. ADCn_SCANMASK selects six of these channels for inclusion in the scan. When an ADC scan is initiated with this configuration, the ADC begins at SCANINPUTID0 and converts each enabled channel in turn. This scan configuration results in a set of six single-ended ADC conversions: PF0, PF3, PA5, PA5, PF7, and PF4.

SCANINPUTSEL	APORT1CH16TO23								APORT1CH16TO23								APORT4CH8TO15								APORT1CH16TO23							
APORT-Channel	1-23	1-22	1-21	1-20	1-19	1-18	1-17	1-16	1-23	1-22	1-21	1-20	1-19	1-18	1-17	1-16	4-15	4-14	4-13	4-12	4-11	4-10	4-9	4-8	1-23	1-22	1-21	1-20	1-19	1-18	1-17	1-16
I/O Pin	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	none	none	PA5	PA4	PA3	PA2	PA1	PA0	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
SCANMASK	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1
SCANINPUTID	31							24	23							16	15							8	7							0

Figure 28.6. ADC Single-ended Scan Mode Example

In differential mode, the default operation of the ADC scanner is to perform a differential measurement between the selected APORT channel and the next channel on that APORT. For example, if the enabled SCANINPUTID points to APORT1CH6, the ADC will perform a differential conversion between APORT1CH6 and APORT1CH7.

There are two exceptions to this rule, listed in order of precedence:

1. When converting SCANINPUTID15, the differential conversion will be performed between the channel selected by SCANINPUTID15 and the channel selected by SCANINPUTID8.
2. When APORTnCH31 is the selected input, the differential conversion will be performed between APORTnCH31 and APORTnCH0.

Figure 28.7 ADC Differential Scan Mode Example on page 1020 shows an example of a differential scan configuration. In this example, ADCn_SCANINPUTSEL has been configured to place APORT1CH16TO23 in the first, third, and fourth channel groups. APORT4CH8TO15 has been placed in the second channel group. ADCn_SCANMASK selects three channels pairs for inclusion in the scan. When an ADC scan is initiated with this configuration, the ADC begins at SCANINPUTID0 and converts each enabled channel in turn. This scan configuration results in a set of three differential ADC conversions: PF0-PF1, PF2-PF3, and PA4-PA5.

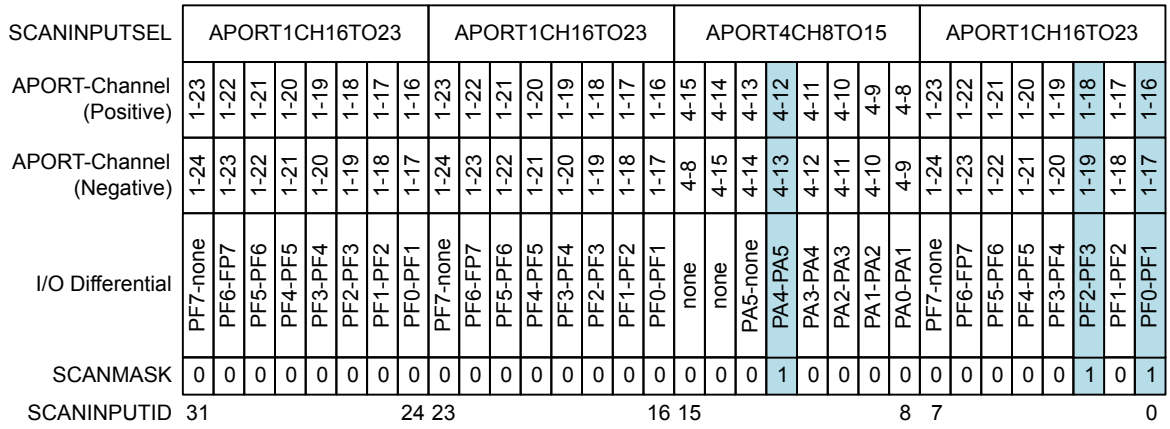


Figure 28.7. ADC Differential Scan Mode Example

In certain applications it may be desirable to perform differential conversions on several channels against a common voltage. The ADCn_SCANNEGSEL register allows eight of the SCANINPUTIDs to re-map the negative terminal of a differential conversion to a common channel. In the first ADCn_SCANINPUTSEL group, the negative input for SCANINPUT 0, 2, 4, and 6 may be re-mapped to any of the odd-numbered channels in that group (SCANINPUT 1, 3, 5, or 7). Likewise, in the second ADCn_SCANINPUTSEL group, the negative input for SCANINPUT 9, 11, 13, and 15 may be re-mapped to any of the even-numbered channels in that group (SCANINPUT 8, 10, 12, or 14).

Figure 28.8 ADC Differential Scan Mode Re-mapping Negative Input Selections on page 1020 shows the effects of the ADCn_SCANNEGSEL register on the re-mappable inputs. The left side of the figure shows the default channel mapping, and the right side of the figure shows how ADCn_SCANNEGSEL can be programmed to map the same negative input on up to four channels.

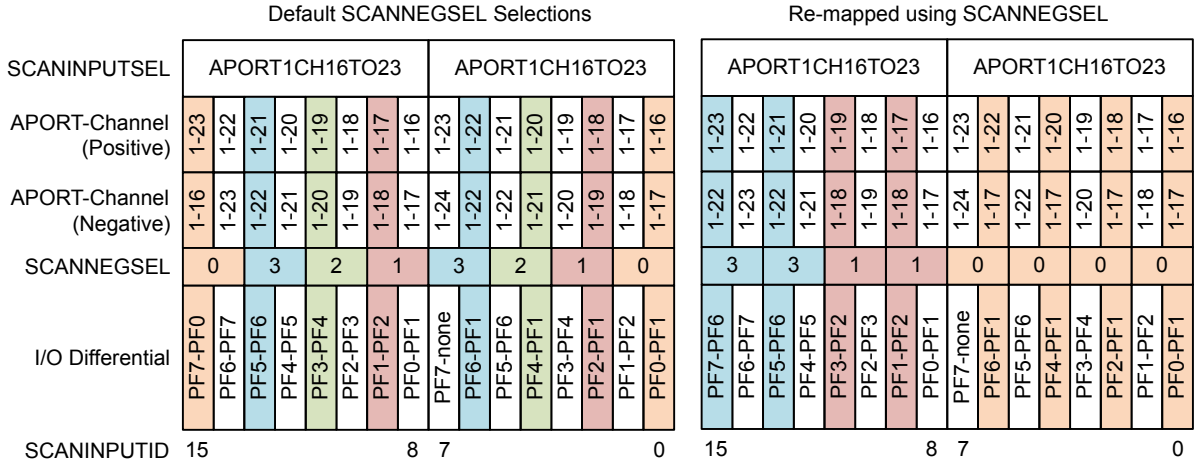


Figure 28.8. ADC Differential Scan Mode Re-mapping Negative Input Selections

28.3.7.3 APORT Conflicts

The ADC shares common analog buses connected to its APORTs (1-4) with other analog peripherals (see device-specific data sheet). As the ADC performs single or scan conversions, it requests the shared buses and sends selections for the control switches to connect the desired I/O pins. If another analog peripheral requests the same shared bus at the same time, there will be a collision and none of the peripherals will be granted control of that bus.

To help debug over-utilization of APORT resources, the ADC hardware provides status information in local registers. The ADCn_APORTREQ register gives the user visibility into which APORT(s) the ADC is requesting given the setting of the input selection registers. ADCn_APORTCONFLICT reports any conflicts that occur. If PROGERR in ADCn_IEN is set, any conflict generates an interrupt. The PROGERR field in the ADCn_STATUS register indicates whether the programming error happened as a result of an APORT bus conflict (BUSCONF) or from a negative-input selection conflict (NEGSELCONF). If the PROGERR interrupt occurred due to a negative selection conflict, then the interrupt can be cleared by software only after correcting the conflict. If a software clear is attempted without correcting the configuration, the interrupt will be cleared for one clock cycle but then it will trigger again as the invalid configuration still persists.

Note: The ADC requests shared bus connections as soon as that bus is selected in the input select registers, even if the ADC is not performing any conversions. This means that by using the APORT request, the ADC will acquire the associated shared analog bus, preventing other peripherals from using it. The bus will be released only when the input select registers are changed.

It is possible for the ADC to passively monitor shared bus signals without controlling the switches and creating bus conflicts. This can be done by setting the ADCn_APORTMASTERDIS register. When ADCn_APORTMASTERDIS is used, channel selection defers to the peripheral acting as the bus master for that shared bus, and no bus conflict will occur. The ADC will connect its input to the shared bus, but the specific channel will be controlled by the peripheral designated as the bus master.

28.3.8 Reference Selection and Input Range Definition

The full scale voltage (VFS) of the ADC is defined as the full input range, from the lowest possible input voltage to the highest. For single-ended conversions, the input range on the selected positive input is from 0 to VFS. For differential conversions, the input to the converter is the difference between the positive and negative input selections. This can range from -VFS/2 to +VFS/2.

VFS for the converter is determined by a combination of the selected voltage reference (VREF) and programmable divider circuits on the ADC input and voltage reference paths. Users have full control over the VREF and divider selections, offering a very flexible and wide selection of VFS values. In most applications however, it is not necessary to adjust VFS beyond a set of common pre-defined choices. For the simplest VFS configuration, refer to 28.3.8.1 Basic Full-Scale Voltage Configuration. If the application requires a VFS configuration not available in the pre-defined choices, 28.3.8.2 Advanced Full-Scale Voltage Configuration covers additional configuration options.

28.3.8.1 Basic Full-Scale Voltage Configuration

Basic configuration of the VFS (full scale voltage) for the converter is done by programming the REF bitfield in ADCn_SINGLECTRL (for single channel mode) or ADCn_SCANCTRL (for scan mode) to any of the pre-defined options. The list of available pre-defined VFS options is:

- VFS = 1.25 V using internal VBGR as the reference source
- VFS = 2.5 V using internal VBGR as the reference source
- VFS = AVDD using AVDD as the reference source (AVDD ≤ 3.6 V)
- VFS = 5 V using internal VBGR as the reference source
- VFS = ADCn_EXTP external pin as a single-ended reference source (1.2 V - 3.6 V)
- VFS = ADCn_EXTP - ADCn_EXTN external pins as a differential reference source. (1.2 V - 3.6 V difference)
- VFS = 2 x AVDD using AVDD as the reference source (AVDD ≤ 3.6 V)

The maximum and minimum input voltage which the ADC can recognize at any external pin is limited to the minimum of the V_{ADC} and IOVDD supply voltages (where V_{ADC} is VDDX_ANA, as described in 28.3.5 Power Supply). If VFS is configured to be larger than the supply range, the full ADC range will not be available. For example, with a 3.3 V supply and VFS configured to 5 V, the input voltage for single-ended conversions will be limited to 0 to 3.3 V, though the effective VFS is still 5 V.

The ADC uses a chip-level bias circuit to provide bias current for its operation. For highest accuracy when using a VBGR-derived internal bandgap reference source, GPBIASACC in ADCn_BIASPROG should be cleared to 0 (HIGHACC). This will allow the ADC to enable high-accuracy mode from the bias circuitry during conversions. When AVDD or an external pin reference option is used, software may set GPBIASACC in ADCn_BIASPROG to 1 (LOWACC) to conserve energy. The GPBIASACC control for all ADCs on chip are considered when selecting bias accuracy, and high accuracy mode takes prioprity. This means that if any ADC has GPBIASACC set to HIGHACC, high accuracy mode will be used for all ADCs. To take advantage of the low-accuracy current savings, GPBIASACC for all ADCs should be set to LOWACC. Note that VDAC and dc-dc usage may also switch the chip-level bias to high- accuracy mode (even if GPBIASACC is set to LOWACC), potentially impacting ADC results. For example, if ADC is doing a conversion with GPBIASACC set to LOWACC and VDAC also starts a conversion using the internal low noise reference, then the chip-level bias circuit will be automatically switched to high-accuracy mode (potentially corrupting results of the on-going ADC conversion). Similarly, dc-dc startup automatically switches the chip-level bias circuit to high-accuracy mode for a short time, i.e., if dc-dc startup happens when ADC is doing a conversion (with GPBIASACC set to LOWACC), ADC results may get corrupted. DC-DC startup automatically switches the bias circuit to high-accuracy mode for 25 µs. It is during this time that ADC conversions with the GPBIASACC set to LOWACC should be avoided.

If the pre-defined VFS options do not suit the particular application, refer to 28.3.8.2 Advanced Full-Scale Voltage Configuration for more advanced VFS options.

28.3.8.2 Advanced Full-Scale Voltage Configuration

For most applications, the pre-defined VFS options described in 28.3.8.1 Basic Full-Scale Voltage Configuration are suitable. Advanced VFS configurations are also possible by programming the REF bitfield in ADCn_SINGLECTRL or ADCn_SCANCTRL to the CONF option. Programming the REF bitfield to CONF allows the user to select the specific VREF source and adjust the programmable input and reference divider options directly.

The general procedure for programming an advanced VFS configuration is as follows:

1. Select the voltage reference source using VREFSEL.
2. Configure VREFATTFIX and VREFATT so that the reference voltage at the ADC is between 0.7 and 1.05 V.
3. Configure VINATT to achieve the desired full-scale voltage.

The VREFSEL field in ADCn_SINGLECTRLX or ADCn_SCANCTRLX selects the voltage reference source. The ADC can choose from the following voltage reference (VREF) sources:

- VBGR: An internal 0.83 V bandgap reference voltage. This is the most precise internal reference source available.
- VDDXWATT: An attenuated version of the AVDD supply voltage. The attenuation factor is determined by the VREFATTFIX and/or VREFATT bit fields.
- VREFPWATT: An external reference source applied to the ADCn_EXTP pin, and attenuated by the attenuation factor (determined by the VREFATTFIX and/or VREFATT bit fields). This is the appropriate choice for external reference inputs greater than 1.05 V.
- VREFP: An external reference source applied to the ADCn_EXTP pin, without any attenuation. This is the appropriate choice for external reference inputs between 0.7 V and 1.05 V.
- VENTROPY: A very low internal reference voltage (approx. 0.1 V). This option is intended to be used only with the ADC inputs tied internally to VSS, for generating random noise at the ADC output.
- VREFPNWATT: A differential version of VREFPWATT, with the reference source applied to the ADCn_EXTP and ADCn_EXTN pins and attenuated. This is the appropriate choice where a differential reference of greater than 1.05 V is required.
- VREFPN: A differential version of VREFP, with the reference source applied to the ADCn_EXTP and ADCn_EXTN pins and no attenuation. This is the appropriate choice where a differential reference of between 0.7 V and 1.05 V is required.
- VBGRLOW: An internal 0.78 V bandgap reference voltage.

The ADC reference voltage should be attenuated to a lower voltage when using AVDD or the external reference source. A simple method for a wide range of reference sources is to set VREFATTFIX to 1. The VREF attenuation factor (ATT_{VREF}) can then be selected between 1/3 (when VREFATT is greater than 0), and 1/4 (when VREFATT is equal to 0). For reference sources between 1.2 V and 3.6 V, ATT_{VREF} = 1/3 is the best choice. ATT_{VREF} = 1/4 can be used with references from 1.6 V to 3.8 V, with slight performance degradation.

Finer granularity on ATT_{VREF} is possible as well, by clearing VREFATTFIX to 0, and setting the VREFATT field. For optimal performance with VREFATTFIX = 0, the attenuated ADC reference input should be limited to between 0.7 V and 1.05 V. When VREFATTFIX is cleared to 0, ATT_{VREF} is set according to the equation:

$$ATT_{VREF} = (VREFATT + 6) / 24 \text{ for } VREFATT < 13, \text{ and } (VREFATT - 3) / 12 \text{ for } VREFATT \geq 13$$

Figure 28.9. ATT_{VREF}: VREF Attenuation Factor

The ADC input also includes a programmable attenuator. The VIN attenuator is used to widen the available input range of the ADC beyond the reference source. The VIN attenuation factor (ATT_{VIN}) is determined by the VINATT field according to the equation:

$$ATT_{VIN} = VINATT / 12 \text{ for } VINATT \geq 3 \text{ (settings 0, 1, and 2 are not allowable values for VINATT)}$$

Figure 28.10. ATT_{VIN}: VIN Attenuation Factor

VFS can be calculated by the formula given below for any given VREF source, VREF attenuation, and VIN attenuation:

$$VFS = 2 \cdot VREF \cdot ATT_{VREF} / ATT_{VIN}$$

VREF is selected in the VREFSEL bitfield, and

ATT_{VREF} is the VREF attenuation factor, determined by VREFATT or VREFATTFIX

ATT_{VIN} is the VIN attenuation factor, determined by VINATT

Figure 28.11. VFS: Full-Scale Input Range

The maximum and minimum input voltage which the ADC can recognize at any external pin is limited to the minimum of the V_{ADC} and IOVDD supply voltages (where V_{ADC} is VDDX_ANA, as described in 28.3.5 Power Supply). If VFS is configured to be larger than the supply range, the full ADC range will not be available. For example, with a 3.3 V supply and VFS configured to 5 V, the input voltage for single-ended conversions will be limited to 0 to 3.3 V, though the effective VFS is still 5 V.

The ADC uses a chip-level bias circuit to provide bias current for its operation. For highest accuracy when using a VBGR-derived internal bandgap reference source, GPBIASACC in ADCn_BIASPROG should be cleared to 0 (HIGHACC). This will allow the ADC to enable high-accuracy mode from the bias circuitry during conversions. When AVDD or an external pin reference option is used, software may set GPBIASACC in ADCn_BIASPROG to 1 (LOWACC) to conserve energy. The GPBIASACC control for all ADCs on chip are considered when selecting bias accuracy, and high accuracy mode takes prioprity. This means that if any ADC has GPBIASACC set to HIGHACC, high accuracy mode will be used for all ADCs. To take advantage of the low-accuracy current savings, GPBIASACC for all ADCs should be set to LOWACC. Note that VDAC and dc-dc usage may also switch the chip-level bias to high- accuracy mode (even if GPBIASACC is set to LOWACC), potentially impacting ADC results. For example, if ADC is doing a conversion with GPBIASACC set to LOWACC and VDAC also starts a conversion using the internal low noise reference, then the chip-level bias circuit will be automatically switched to high-accuracy mode (potentially corrupting results of the on-going ADC conversion). Similarly, dc-dc startup automatically switches the chip-level bias circuit to high-accuracy mode for a short time, i.e., if dc-dc startup happens when ADC is doing a conversion (with GPBIASACC set to LOWACC), ADC results may get corrupted. DC-DC startup automatically switches the bias circuit to high-accuracy mode for 25 µs. It is during this time that ADC conversions with the GPBIASACC set to LOWACC should be avoided.

The combination of VREF, ATT_{VREF} and ATT_{VIN} can produce a wide range of full-scale voltage options for the converter. Table 28.1 Advanced VFS Configuration: VREF = AVDD on page 1024 shows some example VFS configurations using AVDD as a reference source.

Table 28.1. Advanced VFS Configuration: VREF = AVDD

AVDD Voltage	VREF Attenuation Settings	Reference Voltage at ADC	VIN Attenuation Settings	VFS
1.85 V	VREFATTFIX = 0 VREFATT = 6 **ATT_{VREF} = 1/2**	0.925 V	VINATT = 12 **ATT_{VIN} = 1**	1.85 V (+/-0.925 V differential)
3.0 V	VREFATTFIX = 0 VREFATT = 2 **ATT_{VREF} = 1/3**	1.0 V	VINATT = 8 **ATT_{VIN} = 2/3**	3.0 V (+/-1.5 V differential)
3.0 V	VREFATTFIX = 0 VREFATT = 2 **ATT_{VREF} = 1/3**	1.0 V	VINATT = 4 **ATT_{VIN} = 1/3**	6.0 V (+/-3.0 V differential)
3.6 V	VREFATTFIX = 1 VREFATT = 0 **ATT_{VREF} = 1/4**	0.9 V	VINATT = 6 **ATT_{VIN} = 1/2**	3.6 V (+/-1.8 V differential)

28.3.9 Programming of Bias Current

The ADC uses a chip-level bias generator to provide bias current for its operation. The ADC's internal bias can be scaled by ADCBIASPROG field of the ADCn_BIASPROG register. At lower conversion speeds, the ADCBIASPROG can be used to lower active power. Some commonly used settings are given in the ADCBIASPROG register description. For proper operation, the ADC conversion speed must be scaled accordingly. The scale factor is calculated as:

$$\text{Bias scale factor} = (1 - \text{ADCBIASPROG}[2:0]/8) / (1 + 3 \cdot \text{ADCBIASPROG}[3])$$

Figure 28.12. Bias Scale Factor

The bias programming register also includes the VFAULTCLR bit field. If VREFOF interrupt is enabled and it is triggered, then the user needs to set this bit in the ISR before clearing the interrupt flag. This bit then needs to be reset after the interrupt flag is cleared in order to enable the VREFOV flag to trigger on the next VREFOV condition.

The bias current settings should only be changed while the ADC is disabled (i.e. in NORMAL warm-up mode and no conversion in progress).

28.3.10 Feature Set

The following sections explain different ADC features.

28.3.10.1 Conversion Tailgating

Scan conversions have priority over single channel conversions. This means that if scan and single triggers are received simultaneously, or even if the scan is received later when ADC is being warmed up for performing a single conversion, the scan conversion will have priority and will be done before the single conversion. However, a scan trigger will not interrupt in the middle of a single conversion, i.e., if the single conversion is in the acquisition or approximation phase, then the scan will have to wait for the single conversion to complete. If a scan sequence is triggered by a timer on a periodic basis, single channel conversion that started just before a scan trigger can delay the start of the scan sequence, thus causing jitter in sample rate. To solve this, conversion tailgating can be chosen by setting TAILGATE in ADCn_CTRL. When this bit is set, any triggered single channels will wait for the next scan sequence to finish before activating (see Figure 28.13 ADC Conversion Tailgating on page 1025). The single channel will then follow immediately after the scan sequence. In this way, the scan sequence will always start immediately when triggered, provided that the period between the scan triggers is big enough to allow the single sample conversion that was triggered to finish before the next scan trigger arrives. Note that if tailgating is set and a single channel conversion is triggered, it will indefinitely wait for a scan conversion before starting the single channel conversion.

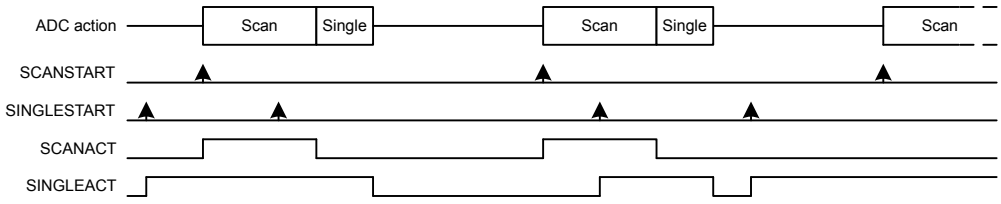


Figure 28.13. ADC Conversion Tailgating

28.3.10.2 Repetitive Mode

Both single channel and scan mode can be run as a one shot conversion or in repetitive mode. The REP bitfield in ADCn_SINGLECTRL/ADCn_SCANCTRL registers can be used to activate the repetitive mode for single and scan respectively. In order to achieve the maximum sampling rate of 1 Msps, repetitive mode should be used.

It is also possible to have a programmable delay between these repetitive conversions. The REPDELAY bitfield in the ADCn_SINGLECTRLX and ADCn_SCANCTRLX registers can be used to set the delay between two repeated conversions in single channel and scan mode respectively. For single channel mode when a single conversion in repetitive mode ends, the user programmed REPDELAY is inserted and then the next single conversion is re triggered after the delay period is over. For scan mode the REPDELAY is inserted after the entire scan sequence ends. Once the delay period is over, scan mode is internally re-triggered. Note that when the ADC is in SYNC mode and REPDELAY is set to generate a delay, it takes an additional 5 HFPERCLK cycles after the trigger before the next conversion begins. If REPDELAY is set to NODELAY, the next conversion begins immediately, without any delay or additional HFPERCLKs. The 28.3.10.1 Conversion Tailgating explains how the single channel and scan mode conversions can push each other out of phase. Conversion tailgating can be chosen in repetitive mode as well in order to ensure that the scan sequence will always start immediately when triggered, provided the scan REPDELAY chosen is big enough for the single conversion to finish. The status flags SINGLEACT and SCANACT stay high throughout the repeat mode, i.e., even during the delay period. The flags show that the conversions are either active or pending. Whether the ADC turns off or stays warmed up between these repeated conversions depends on the WARMUPMODE chosen in the ADCn_CTRL register. When using single channel mode with repeat mode and REPDELAY enabled, then once the ADC has started operation (i.e., singleact status flag has gone high) then no new single conversion triggers (software START/ PRS triggers) should be sent to the ADC until the ADC has stopped converting (i.e., singleact status flag has gone low). The same applies to scan sequence conversions.

28.3.10.3 Conversion Trigger

The conversion modes can be activated by writing a 1 to the SINGLESTART or SCANSTART bit in the ADCn_CMD register. The conversions can be stopped by writing a 1 to the SINGLESTOP or SCANSTOP bit in the ADCn_CMD register. A START command will have priority over a STOP command. When the ADC is stopped in the middle of a conversion, the result buffer is cleared (the FIFO contents for any prior conversions are still intact). Every time a STOP command is issued, the user should wait for the corresponding status flag (SINGLEACT/SCANACT) to go low and then either read all the data in the FIFO or send the corresponding FIFOCLEAR command. The SINGLEACT and SCANACT bits in ADCn_STATUS are set high when the modes are actively converting or have pending conversions.

It is also possible to trigger conversions from PRS signals. The PRS is treated as an asynchronous trigger. Setting PRSEN in ADCn_SINGLECTRL/ADCn_SCANCTRL enables triggering from PRS input. Which PRS channel to listen to is defined by PRSSEL in ADCn_SINGLECTRLX/ADCn_SCANCTRLX. When PRS trigger is selected, it is still possible to trigger a conversion from software. Refer to the PRS chapter for more information on how to set up the PRS channels. When the conversions are triggered using the ADCn_CMD register, then the SINGLEACT and SCANACT bits in the ADCn_STATUS are set as soon as the START command is written to the register. When the conversion is triggered using PRS, it takes some cycles from the time PRS trigger is received until the SINGLEACT and SCANACT bits are set due to the synchronization requirement. If SINGLEACT is already high then sending a new START command or a new PRS trigger for a single conversion will not have any impact as ADC already has a single conversion ongoing or a single conversion pending (single conversion can be pending if ADC is busy running a scan sequence). The same rules apply for SCANACT and SCAN START and PRS triggers. When software issues a SINGLE/SCAN STOP command, it must wait until SINGLEACT/ SCANACT flag goes low before issuing a new START.

The PRS may trigger the ADC in two possible ways, configured by PRSMODE in ADCn_SINGLECTRLX/ADCn_SCANCTRLX. In PULSED mode, a PRS pulse triggers the ADC to start the ADC_CLK (if not already enabled), warm up (if not already warm), start the acquisition period, and perform the conversion. This is identical to issuing a START command from software. In this mode, the input sampling finishes at the end of the acquisition period (AT).

If the ADC_CLK and the source of the trigger (START command or PRS pulse) are not synchronous, the frequency of the input sampling (FS), will experience a $1_{1/2}$ to $2_{1/2}$ ADC_CLK cycle jitter due to synchronization requirements.

To precisely control the sample frequency, the PRSMODE can be set to TIMED mode. In this mode, a long PRS pulse is expected to trigger the ADC and its negative edge directly finishes input sampling and starts the approximation phase, giving precise sampling frequency management. The restriction is that the PRS pulse has to be long enough to start the ADC_CLK (if not already enabled), and finish the acquisition period based on the AT field in ADCn_SINGLECTRL/ADCn_SCANCTRL. The PRS pulse needs to be high when AT event finishes. If it is not high when AT finishes, then it is ignored and input sampling finishes after AT event has ended (a two cycle latency is added to the conversion in this scenario). In this case, the ADC sets the PRSTIMEDERR interrupt flag.

If the PRS pulse is too long (e.g., FS = 32kHz), the analog ADC start can be delayed to save power. The CONVSTARTDELAY along with its EN in the ADCn_SINGLECTRLX or ADCn_SCANCTRLx can be programmed to implement a 0 to 8 microseconds delay. The microsecond tick is counted by TIMEBASE with ADC_CLK similar to warmup case. This saves power as the ADC is not enabled until the last possible microsecond before the fall edge of the PRS arrives to open the sampling switch and to start the approximation phase. Figure 28.14 ADC PRS Timed Mode with ASNEEDED ADC_CLK Request on page 1027) shows PRS Timed mode triggering with CONVSTARTDELAY and ASNEEDED ADC_CLK request. See that power is saved by both delaying the ADC EN and by requesting the ADC_CLK only during ADC operation. This is especially useful in saving power when running the ADC in EM2 DeepSleep or EM3 Stop power mode with low sampling frequency.

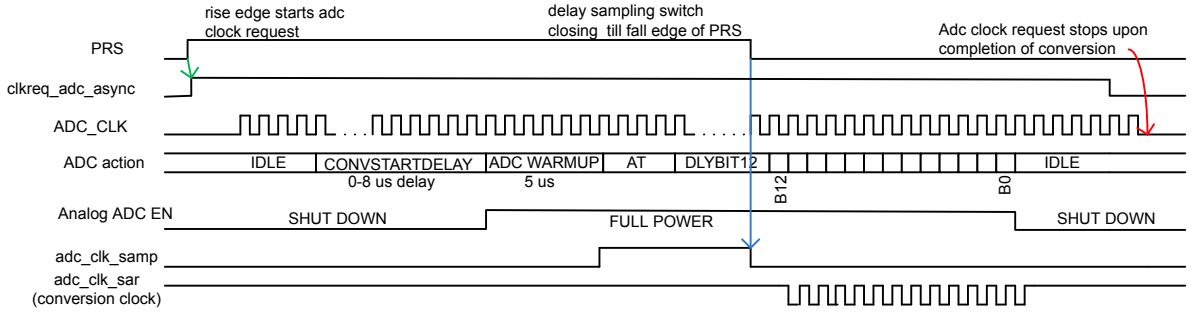


Figure 28.14. ADC PRS Timed Mode with ASNEEDED ADC_CLK Request

When a PRS pulse is received, if the ADC_CLK is not running (ASNEEDED mode), then the ADC requests the clock by setting clkreq_adc_async high. If the chosen clock source (HFXO/ HFSRCCLK/ AUXHFRCO) is already running, then it takes 5 ADC_CLK cycles after the clock request is asserted for the ADC_CLK to start. HFXO and HFSRCCLK (if chosen as ADC clock source) need to be already running before ADC sends out the clock request. If AUXHFRCO is chosen as the ADC clock source, and it is not already

running, then the CMU automatically turns it on when the ADC sends a clock request. In such a case, it takes (7 ADC_CLK cycles + the oscillator startup time) for the ADC_CLK to start. The oscillator startup time can be found in the device data sheet.

When triggering repeat mode using PRS and then stopping the triggered mode using STOP command, ensure that the PRS pulse used to generate the repeat mode has gone low by the time the STOP command is issued. If the PRS pulse continues to stay high after ADC has stopped the ongoing conversion, then it will be picked as a new trigger to start a new conversion.

Note:

- The conversion settings should not be changed while the ADC is running. Doing so may lead to unpredictable behavior.
- The adc_clk_sar phase is always reset by a conversion trigger as long as a conversion is not in progress. This gives predictable latency from the time of the trigger to the time the conversion starts, regardless of when in the trigger occurs.
- Software and LESENSE should not trigger conversions if PRS Timed mode is selected and PRSEN is set to 1 in the ADCn_SINGLECTRL/ADCn_SCANCTRL register.
- If the PRS Timed mode is being used, the acquisition time (AT) must be set greater than 0.

Scan conversions can be triggered using LESENSE as well. LESENSE only triggers one input conversion at a time (not the whole sequence of 32 possible inputs). The input to be converted using LESENSE must be configured by the user in the ADCn_SCANINPUTSEL register before triggering the conversion, i.e., one of the 32 inputs chosen in the ADCn_SCANINPUTSEL register must be the one that is to be converted using LESENSE. The ADCn_SCANMASK is not used for LESENSE triggered conversions. Instead, the user can select which input should be converted through LESENSE inside the LESENSE settings (LESENSE_CHX). The results of LESENSE triggered conversions are not loaded in the FIFO/ DATA registers but are instead available in the LESENSE register. Similarly, the SCAN interrupt flag is not set on completion of a LESENSE triggered conversion (because that flag is set only when the data is written to the Scan FIFO). When there is a LESENSE triggered conversion going on or pending, the SCANACT status flag is set. The SCANPEND interrupt flag is set when a software/PRS triggered scan goes pending because a LESENSE triggered scan is running (software/PRS triggered scan will start after the currently running LESENSE scan conversion completes). Similarly, SCANEXTPEND interrupt flag is set when the LESENSE triggered scan conversion goes pending because a software/PRS triggered scan is running. LESENSE triggered conversions can be stopped at any time using the SCANSTOP command in the ADCn_CMD register. Note that the LESENSE triggered conversion cannot trigger the Scan repeat mode.

The DBGHALT bit-field in the ADCn_CTRL register can be used to choose the ADC behavior in debug mode. If this bit is set to 1, then in debug mode ADC completes the current conversions and then halts. This means that all conversion triggers that were received before the debug halt occurred will be serviced before the ADC halts. All conversion triggers received after the ADC was halted, will be serviced when the debug mode is not halted any more. If the repetitive mode is running (in repetitive mode ADC keeps doing conversions until the user sends a software STOP) and a debug mode halt occurs, then the ADC will gracefully complete the current on-going conversion and then halt. The repetitive mode conversions will restart as soon as the debug mode is not halted any more.

28.3.10.4 Output Results

ADC output results are presented in 2's complement form and the format for single ended and differential conversions are given in Table 28.2 ADC Single Ended Conversion on page 1029 and Table 28.3 ADC Differential Conversion on page 1029, respectively. If differential mode is selected, the results are sign extended up to 32-bits (shown in Table 28.5 ADC Results Representation on page 1030).

Table 28.2. ADC Single Ended Conversion

Input Voltage	Output Results	
	Binary	Hex value
4095/4096 · VFS	111111111111	FFF
0.5 · VFS	100000000000	800
1/4096 · VFS	000000000001	001
0	000000000000	000

Table 28.3. ADC Differential Conversion

Input	Output Results	
	Binary	Hex value
2047/4096 · VFS	011111111111	7FF
0.25 · VFS	010000000000	400
1/4096 · VFS	000000000001	001
0	000000000000	000
-1/4096 · VFS	111111111111	FFF
-0.25 · VFS	110000000000	C00
-0.5 · VFS	100000000000	800

28.3.10.5 Resolution

The ADC performs 12-bit conversions by default. However, if full 12-bit resolution is not needed, it is possible to speed up the conversion by selecting a lower resolution (6 or 8 bits). For more information on the accuracy of the ADC, the reader is referred to the electrical characteristics section for the device.

28.3.10.6 Oversampling

To achieve higher accuracy, hardware oversampling can be enabled individually for each mode (Set RES in ADCn_SINGLECTRL/ADCn_SCANCTRL to 0x3). The oversampling rate (OVSRSEL in ADCn_CTRL) can be set to any integer power of 2 from 2 to 4096 and the configuration is shared between the scan and single channel mode (OVSRSEL field in ADCn_CTRL).

With oversampling, each input is sampled at 12-bits of resolution a number of times (given by OVSRSEL), and the results are filtered by a first order accumulate and dump filter to form the end result. The data presented in the ADCn_SINGLEDATA and ADCn_SCANDATA registers are the direct contents of the accumulation register (sum of samples). However, if the oversampling ratio is set higher than 16x, the accumulated results are shifted to fit the MSB in bit 15 as shown in Table 28.4 Oversampling Result Shifting and Resolution on page 1030.

Table 28.4. Oversampling Result Shifting and Resolution

Oversampling setting	# right shifts	Result Resolution # bits
2x	0	13
4x	0	14
8x	0	15
16x	0	16
32x	1	16
64x	2	16
128x	3	16
256x	4	16
512x	5	16
1024x	6	16
2048x	7	16
4096x	8	16

28.3.10.7 Adjustment

By default, all results are right adjusted, with the LSB of the result in bit position 0 (zero). In differential mode the signed bit is extended up to bit 31, but in single ended mode the bits above the result are read as 0. By setting ADJ in ADCn_SINGLECTRL/ADCn_SCANCTRL, the results are left adjusted as shown in Table 28.5 ADC Results Representation on page 1030. When left adjusted, the MSB is always placed on bit 15 and sign extended to bit 31. All bits below the conversion result are read as 0 (zero).

Table 28.5. ADC Results Representation

Adjustment	Resolution	Bits																
		31 ... 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Right	12	11 ... 11	11	11	11	11	11	10	9	8	7	6	5	4	3	2	1	0
	8	7 ... 7	7	7	7	7	7	7	7	7	7	6	5	4	3	2	1	0
	6	5 ... 5	5	5	5	5	5	5	5	5	5	5	5	4	3	2	1	0
	OVS	15 ... 15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Left	12	11 ... 11	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-
	8	7 ... 7	7	6	5	4	3	2	1	0	-	-	-	-	-	-	-	-
	6	5 ... 5	5	4	3	2	1	0	-	-	-	-	-	-	-	-	-	-
	OVS	15 ... 15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

28.3.10.8 Channel Connection

The inputs are connected to the analog ADC at the beginning of the acquisition phase and are disconnected at the end of the acquisition phase. The time when the APORT switches are closed (for the next input to be converted) can be controlled by the CHCONMODE bitfield in the ADCn_CTRL register. By default, this field is set to the MAXSETTLE option. For MAXSETTLE, APORT switches are closed on the next input as soon as the acquisition phase for the current conversion is complete. This means that the APORT switches are closed approximately 12 adc_clk_sar cycles (assuming 12 bit resolution) before the acquisition phase of the current conversion starts, giving APORT switches maximum time to settle. The time for which APORT switches should be closed before the acquisition phase starts, should be the same for all inputs in order to get consistent results. This means that if the ADC is warmed up with CHCONREFWARMIDLE set to 0 (scan reference warmed up and the APORT switches for the first scan channel closed) and a single trigger comes in, the single conversion will have to wait 12 adc_clk_sar cycles before it can start (even if single is using the same reference as scan). In this case, it might be more suitable to switch to the MAXRESP option in the CHCONMODE bitfield. In MAXRESP, the APORT switches for the upcoming conversion are closed just before the acquisition phase starts. This gives less settling time to the APORT switches but removes the extra waiting time before a conversion can start (which could be the case with MAXSETTLE as discussed above).

28.3.10.9 Temperature Measurement

The ADC includes an internal temperature sensor. This sensor is measured during production test and the temperature readout from the ADC at production temperature, ADC0CAL3_TEMPREAD1V25, is given in the Device Information (DI) page. The production temperature, CAL_TEMP, is also given in this page. The temperature sensor slope, V_TS_SLOPE (mV/degree Celsius), for the sensor is found in the data sheet for the device. Using the 1.25 V VFS option and 12-bit resolution, the temperature can be calculated according to the following formula (VFS in the formula is 1250 mV) :

$$T_{CELSIUS} = CAL\_TEMP - (ADC0CAL3\_TEMPREAD1V25 - ADC\_result) \cdot VFS / (4096 \cdot V\_TS\_SLOPE)$$

Figure 28.15. ADC Temperature Measurement

Note:

- The minimum acquisition time for the temperature reference is found in the electrical characteristics for the device. If using the 1.25 V reference, extra acquisition time is required. In this case the AT field of ADCn_SINGLECTRL or ADCn_SCANCTRL should be set to a value of 9 or higher.
- For the most accurate temperature sensor results, GPBIASACC in ADCn_BIASPROG should be set to 0 to keep the bias in HIGHACC mode.
- If the device has more than one ADC, all ADCs may not be equipped with the temperature sensor. See the device data sheet.

28.3.10.10 ADC as a Random Number Generator

The ADC can be used as a random number generator. This is done by:

1. Choose the REF in the ADCn_SINGLECTRL as CONF, setting the VREFSEL in the ADCn_SINGLECTRLX as VENTROPY and VINATT in the same register to its maximum value of 15.
2. Set DIFF to 1 and RES to 0 in the ADCn_SINGLECTRL register.
3. Trigger a single channel conversion and then read ADCn_SINGLEDATA register when the conversion finishes.

The LSB[2:0] of each sample will be a random number. In this mode, the POSSEL or NEGSEL in ADCn_SINGLECTRL can be connected to VSS or any other noisy input.

28.3.11 Interrupts, PRS Output

The single and scan modes have separate SINGLE and SCAN interrupt flags indicating whether corresponding FIFO contains DVL # of valid conversion data. Corresponding interrupt enable bit has to be set in ADCn_IEN in order to generate interrupts. For these interrupts, there is no software clear mechanism by writing to ADCn_IFC. The user needs to read enough data from the interrupted FIFO to ensure it contains less than DVL # of elements. The ADCn_SINGLEFIFOCOUNT/ADCn_SCANFIFOCOUNT can provide number of valid elements remaining in corresponding FIFO. The FIFO can also be cleared by ADCn_SINGLEFIFOCLEAR/ADCn_SCANFIFOCLEAR, but any existing data will be lost by this operation.

In addition to the SINGLE and SCAN interrupt flags, there is separate scan and single channel result overflow interrupt flag which signals that a result from a scan or single channel FIFO has been overwritten before being read. There is also separate scan and single channel result underflow interrupt flag which signals that a FIFO read was issued when the FIFO was empty.

There is separate scan and single compare interrupt flag which signals a compare match with latest sample if the CMPEN in ADCn_SINGLECTRL/ADCn_SCANCTRL is enabled.

ADC has two separate PRS outputs, one for single channel and one for scan sequence. A finished conversion results in a one ADC_CLK cycle pulse, which is output to the Peripheral Reflex System (PRS). Note that the PRS pulse for scan is generated once after every channel conversion in the scan sequence.

28.3.12 DMA Request

The ADC has two DMA request lines, SINGLEREQ and SCANREQ, which are set when a single or scan FIFO receives DVL# of samples. The requests are cleared when the corresponding single or scan result register is read and corresponding FIFO count reaches lower than DVL. It also has two additional DMA Single request lines, SINGLESREQ and SCANSREQ, that are set when the corresponding FIFO is not empty.

28.3.13 Calibration

The ADC supports offset and gain calibration to correct errors due to process and temperature variations. This must be done individually for each reference used. For each reference, it needs to be repeated for single-ended, negative single-ended (see 28.3.7 Input Selection for details) and differential measurement. The ADC calibration (ADCn_CAL) register contains register fields for calibrating offset and gain for both single and scan mode. The gain and offset calibration are done in single channel mode, but the resulting calibration values can be used for both single and scan mode.

Gain and offset for various references and modes are calibrated during production and the calibration values for these can be found in the Device Information page. During reset, the gain and offset calibration registers are loaded with the production calibration values for the 1V25 reference. Others can be loaded as needed or the user can perform calibration on the fly using the particular reference and mode to be used and write the result in the ADCn_CAL before starting the ADC conversion with them.

28.3.13.1 Offset Calibration

Offset calibration must be performed prior to gain calibration. Follow these steps for the offset calibration in single mode:

1. Select the desired full scale configuration by setting the REF bit field of the ADCn_SINGLECTRL register.
2. Set the AT bit field of the ADCn_SINGLECTRL register to 16CYCLES.
3. Set the POSSEL and NEGSEL of the ADCn_SINGLECTRL register to VSS, and set the DIFF to 1 for enabling differential input if calibrating for DIFF measurement. During calibration, the ADC samples represent the code coming out of the analog. Thus, since the input voltage is 0, the expected ADC output is 0b100000000000 in differential mode, 0b000000000000 in single-ended mode and 0b111111111111 in negative single-ended mode.
4. A binary search is used to find the offset calibration value. Set the CALEN to 1, and OFFSETINVMODE to 1 (if calibrating for negative single-ended conversion) in the ADCn_CAL register. If user is performing negative single-ended calibration, the SINGLEOFFSETINV provides the offset else SINGLEOFFSET bit provides the offset (for both single-ended and differential offset calibration). Start with 0b0000 (or 0b1111 if doing calibration for differential mode) in SINGLEOFFSET or with 0b1000 in SINGLEOFFSETINV (if calibrating for negative single-ended conversion). Set the SINGLESTART bit in the ADCn_CMD register to perform a 12-bit conversion and read the ADCn_SINGLEDATA register. The offset is (ADCn_SINGLEDATA - expected ADC output). Calculate this and write [3:0] of the result into SINGLEOFFSET or SCANOFFSETINV (if doing negative single-ended conversion). The user repeats till ADCn_SINGLEDATA matches expected ADC output. The ADC has a 8LSB built in negative offset to allow for negative offset correction. So, with default offset value, which corrects for the negative offset, the converted ADCn_SINGLEDATA would match expected ADC output if there were no offset. To get better noise immunity, the sampling phase can be repeated with Oversampling enabled. The result of the binary search is written to the SINGLEOFFSET (or SINGLEOFFSETINV) field of the ADCn_CAL register.

28.3.13.2 Gain Calibration

Offset calibration must be performed prior to gain calibration. The Gain Calibration is done in the following manner:

1. Select an external ADC channel for single channel conversion (a differential channel can also be used).
2. Apply an external voltage on the selected ADC input channel. This voltage should correspond to the top of the ADC input range for the selected reference.
3. Set SINGLEGAIN[6:0] to 64 in the ADCn_CAL and measure gain, repeat gain calibration walking the 1 in SINGLEGAIN[6] to SINGLEGAIN[0] till sampled ADCn_SINGLEDATA matches expected value. This is done by setting CALEN in ADCn_CAL set to 1 and performing single channel, reading in the raw ADC code from the ADCn_SINGLEDATA and comparing it with expected code, i.e. 0b111111111111 for single-ended or differential conversion, and 0b000000000000 for negative single-ended conversion. The target value is ideally the top of the ADC input range, but it is recommended to use a value a couple of LSBs below in order to avoid overshooting. The result of the binary search is written to the SINGLEGAIN field of the ADCn_CAL register.

For the VDD reference and external reference, there is no hardware gain calibration. Calibration can be done by software after taking a sample.

28.3.14 EM2 DeepSleep or EM3 Stop Operation

The ADC can operate in EM2 DeepSleep or EM3 Stop mode. For EM2 DeepSleep or EM3 Stop operation the ADC_CLK must be selected as AUXHFRCO. The section 28.3.1 Clock Selection describes how to choose AUXHFRCO as the ADC_CLK. The AUXHFRCO can be kept on for as long as sample conversion is needed or it can be requested by trigger event and after the conversion is done, the AUXHFRCO can be shut down. The second option saves power at the expense of the delay to start the AUXHFRCO oscillator. All the trigger modes are available in EM2 DeepSleep or EM3 Stop as well.

While in EM2 DeepSleep or EM3 Stop, the ADC can wake the system to EM0 Active on enabled interrupts. Following interrupts can wake up the system to EM0 Active:

- SINGLE or SCAN interrupt indicating that the corresponding FIFO has reached the DVL watermark.
- Overflow interrupt (SINGLEOF or SCANOF)
- Underflow interrupt (SINGLEUF or SCANUF), triggered if DMA pops more data than present in the FIFO while the system is asleep
- Compare interrupt (SINGLECMP or SCANCMP)
- Over voltage interrupt (VREFOV)

The ADC can also work with the DMA so that the system does not have to wake up to consume data. This can happen if the SCAN or SINGLE interrupt is disabled and the SINGLEDMAWU or SCANDMAWU in the ADCn_CTRL is set. The DMA will be triggered by the ADC when DVL samples become available in the corresponding FIFO. The DMA will then pop all the elements of the corresponding FIFO and put the system back into the low power state. A system-level wake up will occur upon the DMA done interrupt. Note that other enabled ADC interrupts can still wake up the system when operating with the DMA. For example, the user can configure the window compare function to trip when the result reaches a certain threshold while gathering ADC data in EM2 DeepSleep or EM3 Stop.

The ADC works with the EMU to wake up the system or the DMA. It takes 2 µs from the time the ADC request a wakeup to start of the peripheral clocks. In this ASYNC mode of ADC_CLK, it takes 6 HFPERCCLK cycles to read a single entry from the single or scan FIFO. So, with a 20MHz HFPERCCLK, it takes about 4 µs per DMA wakeup to empty a full FIFO (4 entries). This restricts the sampling rate in EM2 DeepSleep or EM3 Stop in order to avoid FIFO overflows.

The AUXHFRCO power can be reduced by reducing the clock speed, and the user may adjust the ADCBIASPROG field in the ADCn_BIASPROG register to reduce active power of the ADC during the conversions, thus reducing power even more in EM2 DeepSleep/EM3 Stop. Refer to the data sheet for relevant power consumption numbers.

If the ADC is not to be used in EM2 DeepSleep or EM3 Stop, then the user should ensure that the ADC is not busy before going to the low power mode. 28.3.17 ADC Programming Model explains how to ensure the ADC is not busy. If the chip enters EM2 DeepSleep or EM3 Stop when ADC is busy without using AUXHFRCO, then the ADC clock will stop but the ADC will stay on, resulting in higher supply current. If this occurs, the EM23ERR interrupt flag will be set. Software will see this interrupt flag only when the chip wakes up.

28.3.15 ASYNC ADC_CLK Usage Restrictions and Benefits

When the ADC_CLK is chosen to come from ASYNCCLK, (ADCCLKMODE is set to ASYNC), the ADC_CLK and the ADC peripheral clock are considered asynchronous and this adds some restrictions:

- Due to a synchronization delay, accessing the following registers takes extra time (up to additional 7 HFPERCCLK cycles): ADCn_SINGLEDATA, ADCn_SCANDATA, ADCn_SINGLEDATAP, ADCn_SCANDATAP, ADCn_SCANDATAX, ADCn_SCANDATAXP, ADCn_SINGLEFIFOCOUNT, ADCn_SCANFIFOCOUNT, ADCn_SINGLEFIFOCLEAR, ADCn_SCANFIFOCLEAR.
- The safe time to change the ADCn_SINGLECTRL, ADCn_SINGLECTRLX, ADCn_SCANCTRL, ADCn_SCANCTRLx, ADCn_SCANINPUTSEL, ADCn_SCANNEGSEL or ADCn_SCANMASK register is when SINGLEACT/SCANACT in the ADCn_STATUS is 0 with no pending trigger event. The user can enforce this by writing the SINGLESTOP or SCANSTOP in the ADCn_CMD register and ensuring no trigger event can come before modifying the registers.
- When the ADC needs to run in EM2 DeepSleep or EM3 Stop, only AUXHFRCO can provide the ADC_CLK to the ADC. Thus the user needs to set ASYNC mode of ADCCLKMODE and setup the CMU to provide the AUXHFRCO clock as ASYNCCLK.
- If the ADC needs to run on a particular adc_clk_sar frequency to achieve a sample rate and the HFPERCCLK is not a proper multiple for such clock frequency, a higher frequency system clock, HFRCO, can be chosen to be ADC_CLK using ASYNC mode. This allows HFPERCCLK to be set to an optimum value from a system view point.
- ASYNC mode can also help with digital noise mitigation as this clock is asynchronous (not balanced) with the system clock. Moreover, the user can use the invert option to invert the source of ASYNCCLK helping in noise mitigation further.
- Whenever ADC is being used in asynchronous mode, then HFPERCLK must be at least 1.5 times higher than the ADC_CLK.
- With ASNEEDED setting for ASYNCCLK request, the ADC_CLK power can be reduced.

28.3.16 Window Compare Function

The ADC supports a window compare function on both the latest single and scan outputs. The compare thresholds, ADGT and ADLT, are defined in the ADCn_CMPTHR register. These are 16-bit values and their format must match the type of conversion (single-ended or differential) the user is trying to compare with. For example, a 12-bit differential conversion is sign extended to 16 bits while a 12-bit single-ended conversion result would get zero padded to 16-bit result before comparing with ADGT and ADLT. If over-sampling is enabled, the conversion result could grow to 16-bits. There is a single set of ADLT and ADGT threshold for both single and scan compare. The user can however enable single or scan compare logic individually by enabling CMPEN in ADCn_SINGLECTRL or ADCn_SCANCTRL register.

The user can perform comparison both within or outside of the window defined by the ADGT and ADLT. If the ADLT is greater than ADGT, the ADC compares if the current sample is within the window. Otherwise, the ADC compares if the current sample is outside of the window.

28.3.17 ADC Programming Model

The ADC configuration registers are considered static and can only be updated when (1) ADC is in SYNC mode and (2) ADC is idle. ADC is considered busy when it is doing conversions (either the SINGLEACT or SCANACT status flag is high) or when it is warmed up (one of the following status flags is high: WARM, SINGLEREFWARM, SCANREFWARM). The following registers are considered ADC configuration registers: CMU_ADCCTRL, ADCn_CTRL, ADCn_SINGLECTRL, ADCn_SINGLECTRLX, ADCn_SCANCTRL, ADCn_SCANCTRLX, ADCn_SCANINPUTSEL, ADCn_SCANNEGSEL, ADCn_IEN, ADCn_BIASPROG, ADCn_SCANMASK, ADCn_CAL and ADCn_CMPTHR.

From reset, the ADC is in SYNC mode by default. The user can program the configuration registers as needed. If PRS is to be used, PRSEN in ADCn_SINGLECTRL/ADCn_SCANCTRL should be set after all other configuration is complete. Once configuration is complete, the ADC is ready to receive triggers. The user must ensure that no LESENSE triggers come in during the time the ADC configuration registers are being updated.

After the ADC has been used to perform conversions, the user must ensure that the ADC is idle before updating the configuration registers. The first step is to ensure that no new triggers (PRS, LESENSE) are being issued. It can take a few cycles from when a trigger is received to when SINGELACT/SCANACT flags go high due to synchronization requirement. If it is unclear when the triggers were issued and if those are under synchronization or not, the user should add a small delay before checking the status flags. If the SINGLEACT/SCANACT status flags are high, the corresponding STOP command should be issued and the user should wait until the SINGLEACT/SCANACT flags go low. If the ADC was warmed up, then the WARMUPMODE should be changed to NORMAL and then the user should wait on WARM, SINGLEREFWARM and SCANREFWARM flags until those go low. Now the ADC is idle.

If both LESENSE scan and PRS/software scan conversions are taking place, then since there are two scans occurring, the SCAN STOP command needs to be issued twice. The user can check the SCANPENDING status flag. If the flag is set then the user needs to send out 2 SCAN STOP commands. After sending out the first SCAN STOP, the user needs to wait until the SCANPENDING flag goes low. Then the second SCAN STOP command should be issued and the user should wait on the SCANACT status flag to go low.

Note:

When switching ADCCLKMODE in the ADCn_CTRL register, use the appropriate sequence below:

- SYNC to ASYNC:
  1. Disable ADC interrupts
  2. Clear the FIFOs
  3. Switch the ADCCLKMODE

     If the ADC is to be used in ASYNC clock mode with WARMUPMODE set to KEEPADCWARM, then both ADCCLKMODE and WARMUPMODE fields in the ADCn_CTRL register should be set to the desired values in the same register write. This will ensure that the ADC power-on sequence is valid.
- ASYNC TO SYNC:
  1. Disable ADC interrupts
  2. Switch the ADCCLKMODE
  3. Clear the FIFOs

The FIFOs are cleared by writing 1 to the ADCn_SCANFIFOCLEAR and ADCn_SINGLEFIFOCLEAR registers.

When switching from ASYNC to SYNC, ensure that the ASYNC clock is turned off before doing the switch.

28.4 Register Map

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

Offset	Name	Type	Description
0x000	ADCn_CTRL	RW	Control Register
0x008	ADCn_CMD	W1	Command Register
0x00C	ADCn_STATUS	R	Status Register
0x010	ADCn_SINGLECTRL	RW	Single Channel Control Register
0x014	ADCn_SINGLECTRLX	RW	Single Channel Control Register Continued
0x018	ADCn_SCANCTRL	RW	Scan Control Register
0x01C	ADCn_SCANCTRLX	RW	Scan Control Register Continued
0x020	ADCn_SCANMASK	RW	Scan Sequence Input Mask Register
0x024	ADCn_SCANINPUTSEL	RW	Input Selection Register for Scan Mode
0x028	ADCn_SCANNEGSEL	RW	Negative Input Select Register for Scan
0x02C	ADCn_CMPTHR	RW	Compare Threshold Register
0x030	ADCn_BIASPROG	RW	Bias Programming Register for Various Analog Blocks Used in ADC Operation
0x034	ADCn_CAL	RW	Calibration Register
0x038	ADCn_IF	R	Interrupt Flag Register
0x03C	ADCn_IFS	W1	Interrupt Flag Set Register
0x040	ADCn_IFC	(R)W1	Interrupt Flag Clear Register
0x044	ADCn_IEN	RW	Interrupt Enable Register
0x048	ADCn_SINGLEDATA	R(a)	Single Conversion Result Data
0x04C	ADCn_SCANDATA	R(a)	Scan Conversion Result Data
0x050	ADCn_SINGLEDATAP	R	Single Conversion Result Data Peek Register
0x054	ADCn_SCANDATAP	R	Scan Sequence Result Data Peek Register
0x068	ADCn_SCANDATAX	R(a)	Scan Sequence Result Data + Data Source Register
0x06C	ADCn_SCANDATAXP	R	Scan Sequence Result Data + Data Source Peek Register
0x07C	ADCn_APORTREQ	R	APORT Request Status Register
0x080	ADCn_APORTCONFLICT	R	APORT Conflict Status Register
0x084	ADCn_SINGLEFIFOCOUNT	R	Single FIFO Count Register
0x088	ADCn_SCANFIFOCOUNT	R	Scan FIFO Count Register
0x08C	ADCn_SINGLEFIFOCLEAR	W1	Single FIFO Clear Register
0x090	ADCn_SCANFIFOCLEAR	W1	Scan FIFO Clear Register
0x094	ADCn_APORTMASTERDIS	RW	APORT Bus Master Disable Register

28.5 Register Description

28.5.1 ADCn_CTRL - 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		0	0	0x0					0x1F								0x00							0	0		0	0	0	0x0	
Access	RW		RW	RW	RW					RW								RW							RW	RW		RW	RW	RW	RW	
Name	CHCONREFWARMIDLE		CHCONMODE	DBGHALT	OVSRSEL					TIMEBASE								PRESC							ADCCLKMODE	ASYNCCLKEN		TAILGATE	SCANDMAWU	SINGLEDMAWU	WARMUPMODE	

Bit	Name	Reset	Access	Description
31:30	CHCONREFWARMIDLE	0x0	RW	**Channel Connect and Reference Warm Sel When ADC is IDLE**

Channel connect and reference warm preference

Value	Mode	Description
0	PREFSCAN	Keep scan reference warm and APORT switches for first scan channel closed if WARMUPMODE is not NORMAL
1	PREFSINGLE	Keep single reference warm and keep APORT switches for single channel closed if WARMUPMODE is not NORMAL
2	KEEPPREV	Keep last used reference warm and keep APORT switches for corresponding channel closed if WARMUPMODE is not NORMAL

Bit	Name	Reset	Access	Description
29	CHCONMODE	0	RW	**Channel Connect**

Selects Channel Connect Mode

Value	Mode	Description
0	MAXSETTLE	Connect APORT switches for the next input as soon as possible. This optimizes settling time.
1	MAXRESP	Connect APORT switches for the next input at the end of the conversion.

Bit	Name	Reset	Access	Description
28	DBGHALT	0	RW	**Debug Mode Halt Enable**

Selects ADC behavior during debug mode.

Value	Description
0	Continue operation as normal during debug mode.
1	Complete the current conversion and then halt during debug mode.

Bit	Name	Reset	Access	Description
27:24	OVSRSEL	0x0	RW	**Oversample Rate Select**

Select oversampling rate. Oversampling must be enabled for this setting to take effect.

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	X2		2 samples for each conversion result
	1	X4		4 samples for each conversion result
	2	X8		8 samples for each conversion result
	3	X16		16 samples for each conversion result
	4	X32		32 samples for each conversion result
	5	X64		64 samples for each conversion result
	6	X128		128 samples for each conversion result
	7	X256		256 samples for each conversion result
	8	X512		512 samples for each conversion result
	9	X1024		1024 samples for each conversion result
	10	X2048		2048 samples for each conversion result
	11	X4096		4096 samples for each conversion result
23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:16	TIMEBASE	0x1F	RW	**1us Time Base**
	Sets the time base used for the ADC warm up sequence based on ADC_CLK. The TIMEBASE field should be set equal to produce timing of 1us or greater.			
	Value			Description
	TIMEBASE			ADC STANDBY/SLOWACC mode warm-up is set to 1 x (TIMEBASE + 1) ADC_CLK cycles and NORMAL mode warm-up is set to 5 x (TIMEBASE + 1) ADC_CLK cycles.
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	PRESC	0x00	RW	**Prescalar Setting for ADC Sample and Conversion Clock**
	Sets the prescale factor to generate the ADC conversion clock (adc_sar_clk) from ADC_CLK.			
	Value			Description
	PRESC			Clock prescale factor. ADC_CLK is divided by (PRESC+1) to produce adc_clk_sar.
7	ADCCLKMODE	0	RW	**ADC Clock Mode**
	Selects ADC_CLK source as synchronous or asynchronous - with respect to the Peripheral Clock (HFPERCCLK).			
	Value	Mode		Description
	0	SYNC		Synchronous clocking. Uses HFPERCCLK to generate ADC_CLK, ADC will not be available in EM2 in this mode.
	1	ASYNC		Asynchronous clocking. Uses clk_adc_async coming from CMU to generate ADC_CLK. ADC might be available in EM2 in this mode if the CLK_ADC_ASYNC is available in EM2

Bit	Name	Reset	Access	Description
6	ASYNCCLKEN	0	RW	**Selects ASYNC CLK Enable Mode When ADCCLKMODE=1**

Write a 1 to keep ASYNC CLK always enabled.

Value	Mode	Description
0	ASNEEDED	ASYNC CLK is enabled only during ADC Conversion.
1	ALWAYSON	ASYNC CLK is always enabled.

5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	TAILGATE	0	RW	**Conversion Tailgating**

Enable/disable conversion tailgating. Single channel conversions wait for a scan sequence to finish before starting.

Value	Description
0	Scan sequence has priority, but can be delayed by ongoing single channels.
1	Scan sequence has priority and single channels will only start immediately after completion of a scan sequence.

3	SCANDMAWU	0	RW	**SCANFIFO DMA Wakeup**

Selects whether to wakeup the DMA controller when in EM2 and DVL is reached in SCANFIFO

Value	Description
0	While in EM2, the DMA controller will not get requests about DVL reached in SCANFIFO
1	DMA is available in EM2 for processing SCANFIFO DVL request

2	SINGLEDMAWU	0	RW	**SINGLEFIFO DMA Wakeup**

Selects whether to wakeup the DMA controller when in EM2 and DVL is reached in SINGLEFIFO

Value	Description
0	While in EM2, the DMA controller will not get requests about Data Valid Level (DVL) reached in SINGLEFIFO
1	DMA is available in EM2 for processing SINGLEFIFO DVL request

1:0	WARMUPMODE	0x0	RW	**Warm-up Mode**

Select Warm-up Mode for ADC

Value	Mode	Description
0	NORMAL	ADC is shut down after each conversion. 5us warmup time is used before each conversion.
1	KEEPINSTANDBY	ADC is kept in standby mode between conversions. 1us warmup time is used before each conversion.
2	KEEPINSLOWACC	ADC is kept in slow acquisition mode between conversions. 1us warmup time is used before each conversion.
3	KEEPADCWARM	ADC is kept on after conversions, allowing for continuous conversion.

28.5.2 ADCn_CMD - Command 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																													0	0	0	0
Access																													W1	W1	W1	W1
Name																													SCANSTOP	SCANSTART	SINGLESTOP	SINGLESTART

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	SCANSTOP	0	W1	**Scan Sequence Stop**
	Write a 1 to stop scan sequence.			
2	SCANSTART	0	W1	**Scan Sequence Start**
	Write a 1 to start scan sequence.			
1	SINGLESTOP	0	W1	**Single Channel Conversion Stop**
	Write a 1 to stop single channel conversions.			
0	SINGLESTART	0	W1	**Single Channel Conversion Start**
	Write to 1 to start converting in single channel mode.			

28.5.3 ADCn_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															0	0				0	0x0		0	0						0	0	0
Access															R	R				R	R		R	R						R	R	R
Name															SCANDV	SINGLEDV				WARM	PROGERR		SCANREFWARM	SINGLEREFWARM						SCANPENDING	SCANACT	SINGLEACT

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	SCANDV	0	R	**Scan Data Valid**
	SCANCTRLX_DVL # of scan conversion data results are available in Scan FIFO.			
16	SINGLEDV	0	R	**Single Channel Data Valid**
	SINGLECTRLX_DVL # of single channel conversion results are available in Single FIFO.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	WARM	0	R	**ADC Warmed Up**
	ADC is warmed up.			
11:10	PROGERR	0x0	R	**Programming Error Status**
	Programming Error Status			

Mode	Value	Description
BUSCONF	x1	APORT reported a BUS Conflict.
NEGSELCONF	1x	SINGLECTRL's NEGSEL choice is invalid with respect to POSSEL choice. Occurs when two X channels or two Y channels are selected.

9	SCANREFWARM	0	R	**Scan Reference Warmed Up**
	Reference selected for scan mode is warmed up.			
8	SINGLEREFWARM	0	R	**Single Channel Reference Warmed Up**
	Reference selected for single channel mode is warmed up.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	SCANPENDING	0	R	**Scan Conversion Pending**
	Indicates that either an external scan (e.g., lesense triggered) or a PRS/software triggered scan has gone pending. SCANPENDIF and SCANEXTPENDIF show which one of two went pending.			
1	SCANACT	0	R	**Scan Conversion Active**
	Scan sequence is active or has pending conversions.			

Bit	Name	Reset	Access	Description
0	SINGLEACT	0	R	**Single Channel Conversion Active**
	Single channel conversion is active or has pending conversions.			

28.5.4 ADCn_SINGLECTRL - Single Channel 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	0		0		0x0				0xFF								0xFF								0x0			0x0		0	0	0
Access	RW		RW		RW				RW								RW								RW			RW		RW	RW	RW
Name	CMPEN		PRSEN		AT				NEGSEL								POSSEL								REF			RES		ADJ	DIFF	REP

Bit	Name	Reset	Access	Description
31	CMPEN	0	RW	**Compare Logic Enable for Single Channel**

Enable/disable Compare Logic

Value	Description
0	Disable Compare Logic.
1	Enable Compare Logic.

30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	PRSEN	0	RW	**Single Channel PRS Trigger Enable**

Enabled/disable PRS trigger of single channel.

Value	Description
0	Single channel is not triggered by PRS input.
1	Single channel is triggered by PRS input selected by PRSSEL.

28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:24	AT	0x0	RW	**Single Channel Acquisition Time**

Select the acquisition time for single channel.

Value	Mode	Description
0	1CYCLE	1 conversion clock cycle acquisition time for single channel
1	2CYCLES	2 conversion clock cycles acquisition time for single channel
2	3CYCLES	3 conversion clock cycles acquisition time for single channel
3	4CYCLES	4 conversion clock cycles acquisition time for single channel
4	8CYCLES	8 conversion clock cycles acquisition time for single channel
5	16CYCLES	16 conversion clock cycles acquisition time for single channel
6	32CYCLES	32 conversion clock cycles acquisition time for single channel
7	64CYCLES	64 conversion clock cycles acquisition time for single channel
8	128CYCLES	128 conversion clock cycles acquisition time for single channel
9	256CYCLES	256 conversion clock cycles acquisition time for single channel

Bit	Name	Reset	Access	Description
23:16	NEGSEL	0xFF	RW	**Single Channel Negative Input Selection**

Selects the negative input to the ADC for Single Channel Differential mode (in case of singled ended mode, the negative input is grounded). The user can choose any of the 32 channels of any of the 5 BUSes but must ensure that POSSEL and NEGSEL are chosen from different resources (X or Y) BUS. In case of an invalid configuration, the ADC will perform a single-ended sampling and issue a BUSCONFLICT IRQ.

Mode	Value	Description
APORT0XCH0	0	Select APORT0XCH0
APORT0XCH1	1	Select APORT0XCH1
...	...	.....
APORT0XCH15	15	Select APORT0XCH15
APORT0YCH0	16	Select APORT0YCH0
APORT0YCH1	17	Select APORT0YCH1
APORT0YCH15	31	Select APORT0YCH15
APORT1XCH0	32	Select APORT1XCH0
APORT1YCH1	33	Select APORT1YCH1
...	...	.....
APORT1YCH31	63	Select APORT1YCH31
APORT2YCH0	64	Select APORT2YCH0
APORT2XCH1	65	Select APORT2XCH1
...	...	.....
APORT2XCH31	95	Select APORT2XCH31
APORT3XCH0	96	Select APORT3XCH0
APORT3YCH1	97	Select APORT3YCH1
...	...	.....
APORT3YCH31	127	Select APORT3YCH31
APORT4YCH0	128	Select APORT4YCH0
APORT4XCH1	129	Select APORT4XCH1
...	...	.....
APORT4XCH31	159	Select APORT4XCH31
TESTN	245	Reserved for future expansion
VSS	255	VSS

15:8	POSSEL	0xFF	RW	**Single Channel Positive Input Selection**

Selects the positive input to the ADC for single channel operation. Software can choose any of the 32 channels of any BUS as positive input. In DIFF mode POSSEL and NEGSEL need to be chosen from different resources (X or Y). If an X BUS is connected to POSSEL, only a Y BUS can connect to NEGSEL, and vice-versa. The user can also select some internal nodes as positive input for single-ended sampling. These internal nodes cannot be sampled differentially.

Mode	Value	Description
APORT0XCH0	0	Select APORT0XCH0
APORT0XCH1	1	Select APORT0XCH1

Bit	Name	Reset	Access	Description
	...	...		.....
	APORT0XCH15	15		Select APORT0XCH15
	APORT0YCH0	16		Select APORT0YCH0
	APORT0YCH1	17		Select APORT0YCH1
	APORT0YCH15	31		Select APORT0YCH15
	APORT1XCH0	32		Select APORT1XCH0
	APORT1YCH1	33		Select APORT1YCH1
	...	...		.....
	APORT1YCH31	63		Select APORT1YCH31
	APORT2YCH0	64		Select APORT2YCH0
	APORT2XCH1	65		Select APORT2XCH1
	...	...		.....
	APORT2XCH31	95		Select APORT2XCH31
	APORT3XCH0	96		Select APORT3XCH0
	APORT3YCH1	97		Select APORT3YCH1
	...	...		.....
	APORT3YCH31	127		Select APORT3YCH31
	APORT4YCH0	128		Select APORT4YCH0
	APORT4XCH1	129		Select APORT4XCH1
	...	...		.....
	APORT4XCH31	159		Select APORT4XCH31
	AVDD	224		Select AVDD
	BUVDD	225		Select BUVDD
	DVDD	226		Select DVDD
	PAVDD	227		Reserved for future use
	DECOUPLE	228		Select DECOUPLE
	IOVDD	229		Select IOVDD
	IOVDD1	230		Select IOVDD1. Not Applicable if no IOVDD1 is available.
	VSP	231		Reserved for future expansion
	OPA2	242		OPA2 output. Not Applicable if no OPA is available.
	TEMP	243		Temperature sensor
	DAC0OUT0	244		DAC0 output 0. Not Applicable if no DAC is available.
	R5VOUT	245		5V sub-system ADC mux output. Not Applicable if no 5V sub-system is available.
	SP1	246		Reserved for future expansion
	SP2	247		Reserved for future expansion
	DAC0OUT1	248		DAC0 output 1. Not Applicable if no DAC is available.

Bit	Name	Reset	Access	Description
	SUBLSB	249		SUBLSB measurement enabled.
	OPA3	250		OPA3 output. Not Applicable if no OPA is available.
	VSS	255		VSS
7:5	REF	0x0	RW	**Single Channel Reference Selection**
	Select reference to ADC single channel mode.			
	Value	Mode		Description
	0	1V25		VFS = 1.25V with internal VBGR reference
	1	2V5		VFS = 2.5V with internal VBGR reference
	2	VDD		VFS = AVDD with AVDD as reference source
	3	5V		VFS = 5V with internal VBGR reference
	4	EXTSINGLE		Single ended external reference
	5	2XEXTDIFF		Differential external reference, 2x
	6	2XVDD		VFS = 2xAVDD with AVDD as the reference source
	7	CONF		Use SINGLECTRLX to configure reference
4:3	RES	0x0	RW	**Single Channel Resolution Select**
	Select single channel conversion resolution.			
	Value	Mode		Description
	0	12BIT		12-bit resolution.
	1	8BIT		8-bit resolution.
	2	6BIT		6-bit resolution.
	3	OVS		Oversampling enabled. Oversampling rate is set in OVSRSEL.
2	ADJ	0	RW	**Single Channel Result Adjustment**
	Select single channel result adjustment.			
	Value	Mode		Description
	0	RIGHT		Results are right adjusted.
	1	LEFT		Results are left adjusted.
1	DIFF	0	RW	**Single Channel Differential Mode**
	Select single ended or differential input.			
	Value			Description
	0			Single ended input.
	1			Differential input.
0	REP	0	RW	**Single Channel Repetitive Mode**
	Enable/disable repetitive single channel conversions.			
	Value			Description

Bit	Name	Reset	Access	Description
	0			ADC will perform one conversion per trigger in single channel mode.
	1			ADC will repeat conversions in single channel mode continuously until SINGLESTOP is written.

28.5.5 ADCn_SINGLECTRLX - Single Channel Control Register Continued

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				0	0x00						0x0				0		0	0x0		0x0				0x0				0	0x0		
Access	RW				RW	RW						RW				RW		RW	RW		RW				RW				RW	RW		
Name	REPDELAY				CONVSTARTDELAYEN	CONVSTARTDELAY						PRSSEL				PRSMODE		FIFOOFACT	DVL		VINATT				VREFATT				VREFATTFIX	VREFSEL		

Bit	Name	Reset	Access	Description
31:29	REPDELAY	0x0	RW	**REPDELAY Select for SINGLE REP Mode**

Delay value between two repeated conversions.

Value	Mode	Description
0	NODELAY	No delay
1	4CYCLES	4 conversion clock cycles
2	8CYCLES	8 conversion clock cycles
3	16CYCLES	16 conversion clock cycles
4	32CYCLES	32 conversion clock cycles
5	64CYCLES	64 conversion clock cycles
6	128CYCLES	128 conversion clock cycles
7	256CYCLES	256 conversion clock cycles

28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	CONVSTARTDELAYEN	0	RW	**Enable Delaying Next Conversion Start**

Delay value for next conversion start event.

Value	Description
0	CONVSTARTDELAY is disabled.
1	CONVSTARTDELAY is enabled.

26:22	CONVSTARTDELAY	0x00	RW	**Delay Value for Next Conversion Start If CONVSTARTDELAYEN is Set**

Delay value for next conversion start event in 1us ticks (based on TIMEBASE).

Value	Description
DELAY	Delay the next conversion start by (CONVSTARTDELAY+1) us

Bit	Name	Reset	Access	Description
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20:17	PRSSEL	0x0	RW	**Single Channel PRS Trigger Select**

Select PRS trigger for single channel.

Value	Mode	Description
0	PRSCH0	PRS ch 0 triggers single channel
1	PRSCH1	PRS ch 1 triggers single channel
2	PRSCH2	PRS ch 2 triggers single channel
3	PRSCH3	PRS ch 3 triggers single channel
4	PRSCH4	PRS ch 4 triggers single channel
5	PRSCH5	PRS ch 5 triggers single channel
6	PRSCH6	PRS ch 6 triggers single channel
7	PRSCH7	PRS ch 7 triggers single channel
8	PRSCH8	PRS ch 8 triggers single channel
9	PRSCH9	PRS ch 9 triggers single channel
10	PRSCH10	PRS ch 10 triggers single channel
11	PRSCH11	PRS ch 11 triggers single channel
12	PRSCH12	PRS ch 12 triggers single channel
13	PRSCH13	PRS ch 13 triggers single channel
14	PRSCH14	PRS ch 14 triggers single channel
15	PRSCH15	PRS ch 15 triggers single channel

16	PRSMODE	0	RW	**Single Channel PRS Trigger Mode**

PRS trigger mode of single channel.

Value	Mode	Description
0	PULSED	Single channel trigger is considered a regular asynchronous pulse that starts ADC warm-up, then acquisition/conversion sequence. The ADC_CLK controls the warmup-time.
1	TIMED	Single channel trigger should be a pulse long enough to provide the required warm-up time for the selected ADC warmup mode. The negative edge requests sample acquisition. DELAY can be used to delay the warm-up request if the pulse is too long.

15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	FIFOOFACT	0	RW	**Single Channel FIFO Overflow Action**

Select how FIFO behaves when full

Value	Mode	Description
0	DISCARD	FIFO stops accepting new data if full, triggers SINGLEOF IRQ.
1	OVERWRITE	FIFO overwrites old data when full, triggers SINGLEOF IRQ.

Bit	Name	Reset	Access	Description
13:12	DVL	0x0	RW	**Single Channel DV Level Select**
	Select single channel Data Valid level. SINGLE IRQ is set when (DVL+1) number of single channels have been converted and their results are available in the Single FIFO.			
11:8	VINATT	0x0	RW	**Code for VIN Attenuation Factor**
	Used to set the VIN attenuation factor.			
7:4	VREFATT	0x0	RW	**Code for VREF Attenuation Factor When VREFSEL is 1, 2 or 5**
	Used to set VREF attenuation factor.			
3	VREFATTFIX	0	RW	**Enable Fixed Scaling on VREF**
	Enables fixed scaling on VREF			

Value	Description
0	VREFATT setting is used to scale VREF when VREFSEL is 1, 2 or 5.
1	A fixed VREF attenuation is used to cover a large reference source range. When VREFATT = 0, the scaling factor is 1/4. For non-zero values of VREFATT, the scaling factor is 1/3.

2:0	VREFSEL	0x0	RW	**Single Channel Reference Selection**
	Select reference VREF to ADC single channel mode.			

Value	Mode	Description
0	VBGR	Internal 0.83V Bandgap reference
1	VDDXWATT	Scaled AVDD: AVDD*(the VREF attenuation factor)
2	VREFPWATT	Scaled singled ended external Vref: ADCn_EXTP*(the VREF attenuation factor)
3	VREFP	Raw single ended external Vref: ADCn_EXTP
4	VENTROPY	Special mode used to generate ENTROPY.
5	VREFPNWATT	Scaled differential external Vref from : (ADCn_EXTP-ADCn_EXTN)*(the VREF attenuation factor)
6	VREFPN	Raw differential external Vref from : (ADCn_EXTP-ADCn_EXTN)
7	VBGRLOW	Internal Bandgap reference at low setting 0.78V

28.5.6 ADCn_SCANCTRL - Scan 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	0		0		0x0																				0x0			0x0		0	0	0
Access	RW		RW		RW																				RW			RW		RW	RW	RW
Name	CMPEN		PRSEN		AT																				REF			RES		ADJ	DIFF	REP

Bit	Name	Reset	Access	Description
31	CMPEN	0	RW	**Compare Logic Enable for Scan**

Enable/disable Compare Logic

Value	Description
0	Disable Compare Logic.
1	Enable Compare Logic.

Bit	Name	Reset	Access	Description
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	PRSEN	0	RW	**Scan Sequence PRS Trigger Enable**

Enabled/disable PRS trigger of scan sequence.

Value	Description
0	Scan sequence is not triggered by PRS input
1	Scan sequence is triggered by PRS input selected by PRSSEL

Bit	Name	Reset	Access	Description
28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27:24	AT	0x0	RW	**Scan Acquisition Time**

Select the acquisition time for scan.

Value	Mode	Description
0	1CYCLE	1 conversion clock cycle acquisition time for scan
1	2CYCLES	2 conversion clock cycles acquisition time for scan
2	3CYCLES	3 conversion clock cycles acquisition time for scan
3	4CYCLES	4 conversion clock cycles acquisition time for scan
4	8CYCLES	8 conversion clock cycles acquisition time for scan
5	16CYCLES	16 conversion clock cycles acquisition time for scan
6	32CYCLES	32 conversion clock cycles acquisition time for scan
7	64CYCLES	64 conversion clock cycles acquisition time for scan
8	128CYCLES	128 conversion clock cycles acquisition time for scan
9	256CYCLES	256 conversion clock cycles acquisition time for scan

Bit	Name	Reset	Access	Description
23:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:5	REF	0x0	RW	**Scan Sequence Reference Selection**

Select reference to ADC scan sequence.

Value	Mode	Description
0	1V25	VFS = 1.25V with internal VBGR reference
1	2V5	VFS = 2.5V with internal VBGR reference
2	VDD	VFS = AVDD with AVDD as reference source
3	5V	VFS = 5V with internal VBGR reference
4	EXTSINGLE	Single ended external reference
5	2XEXTDIFF	Differential external reference, 2x
6	2XVDD	VFS=2xAVDD with AVDD as the reference source
7	CONF	Use SCANCTRLX to configure reference

4:3	RES	0x0	RW	**Scan Sequence Resolution Select**

Select scan sequence conversion resolution.

Value	Mode	Description
0	12BIT	12-bit resolution
1	8BIT	8-bit resolution
2	6BIT	6-bit resolution
3	OVS	Oversampling enabled. Oversampling rate is set in OVSRSEL

2	ADJ	0	RW	**Scan Sequence Result Adjustment**

Select scan sequence result adjustment.

Value	Mode	Description
0	RIGHT	Results are right adjusted
1	LEFT	Results are left adjusted

1	DIFF	0	RW	**Scan Sequence Differential Mode**

Select single ended or differential input.

Value	Description
0	Single ended input
1	Differential input

0	REP	0	RW	**Scan Sequence Repetitive Mode**

Enable/disable repetitive scan sequence.

Value	Description
0	Scan conversion mode is deactivated after one sequence.

Bit	Name	Reset	Access	Description
	1			Scan conversion mode repeats continuously until SCANSTOP is written.

28.5.7 ADCn_SCANCTRLX - Scan Control Register Continued

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				0	0x00						0x0				0		0	0x0		0x0				0x0				0	0x0		
Access	RW				RW	RW						RW				RW		RW	RW		RW				RW				RW	RW		
Name	REPDELAY				CONVSTARTDELAYEN	CONVSTARTDELAY						PRSSEL				PRSMODE		FIFOOFACT	DVL		VINATT				VREFATT				VREFATTFIX	VREFSEL		

Bit	Name	Reset	Access	Description
31:29	REPDELAY	0x0	RW	**REPDELAY Select for SCAN REP Mode**

Delay value between two repeated conversions.

Value	Mode	Description
0	NODELAY	No delay
1	4CYCLES	4 conversion clock cycles
2	8CYCLES	8 conversion clock cycles
3	16CYCLES	16 conversion clock cycles
4	32CYCLES	32 conversion clock cycles
5	64CYCLES	64 conversion clock cycles
6	128CYCLES	128 conversion clock cycles
7	256CYCLES	256 conversion clock cycles

28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	CONVSTARTDELAYEN	0	RW	**Enable Delaying Next Conversion Start**

Delay value for next conversion start event.

Value	Description
0	CONVSTARTDELAY is disabled
1	CONVSTARTDELAY is enabled.

26:22	CONVSTARTDELAY	0x00	RW	**Delay Next Conversion Start If CONVSTARTDELAYEN is Set**

Delay value for next conversion start event in 1us ticks (based on TIMEBASE)

Value	Description
DELAY	Delay the next conversion start by (DELAY+1) us

Bit	Name	Reset	Access	Description
21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20:17	PRSSEL	0x0	RW	**Scan Sequence PRS Trigger Select**

Select PRS trigger for scan sequence.

Value	Mode	Description
0	PRSCH0	PRS ch 0 triggers scan sequence
1	PRSCH1	PRS ch 1 triggers scan sequence
2	PRSCH2	PRS ch 2 triggers scan sequence
3	PRSCH3	PRS ch 3 triggers scan sequence
4	PRSCH4	PRS ch 4 triggers scan sequence
5	PRSCH5	PRS ch 5 triggers scan sequence
6	PRSCH6	PRS ch 6 triggers scan sequence
7	PRSCH7	PRS ch 7 triggers scan sequence
8	PRSCH8	PRS ch 8 triggers scan sequence
9	PRSCH9	PRS ch 9 triggers scan sequence
10	PRSCH10	PRS ch 10 triggers scan sequence
11	PRSCH11	PRS ch 11 triggers scan sequence
12	PRSCH12	PRS ch 12 triggers scan sequence
13	PRSCH13	PRS ch 13 triggers scan sequence
14	PRSCH14	PRS ch 14 triggers scan sequence
15	PRSCH15	PRS ch 15 triggers scan sequence

16	PRSMODE	0	RW	**Scan PRS Trigger Mode**

PRS trigger mode of scan.

Value	Mode	Description
0	PULSED	Scan trigger is considered a regular async pulse that starts ADC warm-up, then acquisition/conversion sequence. The ADC_CLK controls the warmup-time.
1	TIMED	Scan trigger should be a pulse long enough to provide the required warm-up time for the selected ADC warmup mode. The negative edge requests sample acquisition. DELAY can be used to delay the warm-up request if the pulse is too long.

15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	FIFOOFACT	0	RW	**Scan FIFO Overflow Action**

Select how FIFO behaves when full

Value	Mode	Description
0	DISCARD	FIFO stops accepting new data if full, triggers SCANOF IRQ.
1	OVERWRITE	FIFO overwrites old data when full, triggers SCANOF IRQ.

Bit	Name	Reset	Access	Description
13:12	DVL	0x0	RW	**Scan DV Level Select**
	Select Scan Data Valid level. SCAN IRQ is set when (DVL+1) number of scan channels have been converted and their results are available in the SCAN FIFO.			
11:8	VINATT	0x0	RW	**Code for VIN Attenuation Factor**
	Used to set the VIN attenuation factor.			
7:4	VREFATT	0x0	RW	**Code for VREF Attenuation Factor When VREFSEL is 1, 2 or 5**
	Used to set VREF attenuation factor.			
3	VREFATTFIX	0	RW	**Enable Fixed Scaling on VREF**
	Enables fixed scaling on VREF			

Value	Description
0	VREFATT setting is used to scale VREF when VREFSEL is 1, 2 or 5.
1	A fixed VREF attenuation is used to cover a large reference source range. When VREFATT = 0, the scaling factor is 1/4. For non-zero values of VREFATT, the scaling factor is 1/3.

2:0	VREFSEL	0x0	RW	**Scan Channel Reference Selection**
	Select reference VREF to ADC scan channel mode.			

Value	Mode	Description
0	VBGR	Internal 0.83V Bandgap reference
1	VDDXWATT	Scaled AVDD: AVDD*(the VREF attenuation factor)
2	VREFPWATT	Scaled singled ended external Vref: ADCn_EXTP*(the VREF attenuation factor)
3	VREFP	Raw single ended external Vref: ADCn_EXTP
5	VREFPNWATT	Scaled differential external Vref from : (ADCn_EXTP-ADCn_EXTN)*(the VREF attenuation factor)
6	VREFPN	Raw differential external Vref from : (ADCn_EXTP-ADCn_EXTN)
7	VBGRLOW	Internal Bandgap reference at low setting 0.78V

28.5.8 ADCn_SCANMASK - Scan Sequence Input 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	0x00000000																															
Access	RW																															
Name	SCANINPUTEN																															

Bit	Name	Reset	Access	Description
31:0	SCANINPUTEN	0x00000000	RW	**Scan Sequence Input Mask**

Set one or more bits in this mask to select which inputs are included in scan sequence in either single ended or differential mode. This works with SCANINPUTSEL register. The SCANINPUTSEL chooses 32 possible channels for single-ended or 32 pairs of possible channels for differential scanning from BUSes. These chosen channels are referred as ADCn_INPUTx in the description. Four even inputs from first group of 8 ADCn_INPUTx and four odd inputs from second group of 8 ADCn_INPUTx have programmable NEGSEL, defined in SCANNEGSEL register. If the SCANMASK is set to 0 and scan conversion is triggered, ADC will do a conversion with garbage results since no inputs were enabled for conversion.

Mode	Value	Description
DIFF = 0		
INPUT0	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1	ADCn_INPUT0 included in mask
INPUT1	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1x	ADCn_INPUT1 included in mask
INPUT2	xxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xx	ADCn_INPUT2 included in mask
INPUT3	xxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx	ADCn_INPUT3 included in mask
INPUT4	xxxxxxxxxxxxxxxxxxxxxxxxxxx1xxxx	ADCn_INPUT4 included in mask
INPUT5	xxxxxxxxxxxxxxxxxxxxxxxxxx1xxxxx	ADCn_INPUT5 included in mask
INPUT6	xxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx	ADCn_INPUT6 included in mask
INPUT7	xxxxxxxxxxxxxxxxxxxxxxxx1xxxxxxx	ADCn_INPUT7 included in mask
...	.....	.....
INPUT31	1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	ADCn_INPUT31 included in mask
DIFF = 1		
INPUT0INPUT0NEGSEL	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1	(Positive input: ADCn_INPUT0 Negative input: chosen by INPUT0NEGSEL) included in mask

Bit	Name	Reset	Access	Description
	INPUT1INPUT2	xxxxxxxxxxxxxxxxxxxx xxxxxxxxx1x		(Positive input: ADCn_INPUT1 Negative input: ADCn_INPUT2) included in mask
	INPUT2INPUT2NEGSEL	xxxxxxxxxxxxxxxxxxxx xxxxxxxx1xx		(Positive input: ADCn_INPUT2 Negative input: chosen by INPUT2NEGSEL) included in mask
	INPUT3INPUT4	xxxxxxxxxxxxxxxxxxxx xxxxxxx1xxx		(Positive input: ADCn_INPUT3 Negative input: ADCn_INPUT4) included in mask
	INPUT4INPUT4NEGSEL	xxxxxxxxxxxxxxxxxxxx xxxxxx1xxxx		(Positive input: ADCn_INPUT4 Negative input: chosen by INPUT4NEGSEL) included in mask
	INPUT5INPUT6	xxxxxxxxxxxxxxxxxxxx xxxxx1xxxxx		(Positive input: ADCn_INPUT5 Negative input: ADCn_INPUT6) included in mask
	INPUT6INPUT6NEGSEL	xxxxxxxxxxxxxxxxxxxx xxxx1xxxxxx		(Positive input: ADCn_INPUT6 Negative input: chosen by INPUT6NEGSEL) included in mask
	INPUT7INPUT0	xxxxxxxxxxxxxxxxxxxx xxx1xxxxxxx		(Positive input: ADCn_INPUT7 Negative input: ADCn_INPUT8) included in mask
	INPUT8INPUT9	xxxxxxxxxxxxxxxxxxxx xx1xxxxxxxx		(Positive input: ADCn_INPUT8 Negative input: ADCn_INPUT9) included in mask
	INPUT9INPUT9NEGSEL	xxxxxxxxxxxxxxxxxxxx x1xxxxxxxxx		(Positive input: ADCn_INPUT9 Negative input: chosen by INPUT9NEGSEL) included in mask
	INPUT10INPUT11	xxxxxxxxxxxxxxxxxxxx 1xxxxxxxxxx		(Positive input: ADCn_INPUT10 Negative input: ADCn_INPUT11) included in mask
	INPUT11INPUT11NEGSEL	xxxxxxxxxxxxxxxxxxx1 xxxxxxxxxxx		(Positive input: ADCn_INPUT11 Negative input: chosen by INPUT11NEGSEL) included in mask
	INPUT12INPUT13	xxxxxxxxxxxxxxxxxx1x xxxxxxxxxxx		(Positive input: ADCn_INPUT12 Negative input: ADCn_INPUT13) included in mask
	INPUT13INPUT13NEGSEL	xxxxxxxxxxxxxxxxx1xx xxxxxxxxxxx		(Positive input: ADCn_INPUT13 Negative input: chosen by INPUT13NEGSEL) included in mask
	INPUT14INPUT15	xxxxxxxxxxxxxxxx1xxx xxxxxxxxxxx		(Positive input: ADCn_INPUT14 Negative input: ADCn_INPUT15) included in mask
	INPUT15INPUT15NEGSEL	xxxxxxxxxxxxxxx1xxxx xxxxxxxxxxx		(Positive input: ADCn_INPUT15 Negative input: chosen by INPUT15NEGSEL) included in mask
	INPUT16INPUT17	xxxxxxxxxxxxxx1xxxxx xxxxxxxxxxx		(Positive input: ADCn_INPUT16 Negative input: ADCn_INPUT17) included in mask
	.....	.....		.....
	INPUT28INPUT29	xxx1xxxxxxxxxxxxxxxx xxxxxxxxxxx		(Positive input: ADCn_INPUT28 Negative input: ADCn_INPUT29) included in mask
	INPUT29INPUT30	xx1xxxxxxxxxxxxxxxxx xxxxxxxxxxx		(Positive input: ADCn_INPUT29 Negative input: ADCn_INPUT30) included in mask
	INPUT30INPUT31	x1xxxxxxxxxxxxxxxxxx xxxxxxxxxxx		(Positive input: ADCn_INPUT30 Negative input: ADCn_INPUT31) included in mask
	INPUT31INPUT24	1xxxxxxxxxxxxxxxxxxx xxxxxxxxxxx		(Positive input: ADCn_INPUT31 Negative input: ADCn_INPUT24) included in mask

28.5.9 ADCn_SCANINPUTSEL - Input Selection Register for Scan Mode

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				0x00								0x00								0x00								0x00				
Access				RW								RW								RW								RW				
Name				INPUT24TO31SEL								INPUT16TO23SEL								INPUT8TO15SEL								INPUT0TO7SEL				

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	INPUT24TO31SEL	0x00	RW	**Inputs Chosen for ADCn_INPUT24-ADCn_INPUT31 as Referred in SCANMASK**

Mode	Value	Description
APORT0CH0TO7	0	Select APORT0's CH0-CH7 as ADCn_INPUT24-ADCn_INPUT31
APORT0CH8TO15	1	Select APORT0's CH8-CH15 as ADCn_INPUT24-ADCn_INPUT31
APORT1CH0TO7	4	Select APORT1's CH0-CH7 as ADCn_INPUT24-ADCn_INPUT31
APORT1CH8TO15	5	Select APORT1's CH8-CH15 as ADCn_INPUT24-ADCn_INPUT31
APORT1CH16TO23	6	Select APORT1's CH16-CH23 as ADCn_INPUT24-ADCn_INPUT31
APORT1CH24TO31	7	Select APORT1's CH24-CH31 as ADCn_INPUT24-ADCn_INPUT31
APORT2CH0TO7	8	Select APORT2's CH0-CH7 as ADCn_INPUT24-ADCn_INPUT31
...	.	.....
APORT3CH0TO7	12	Select APORT3's CH0-CH7 as ADCn_INPUT24-ADCn_INPUT31
...	.	.....
APORT4CH0TO7	16	Select APORT4's CH0-CH7 as ADCn_INPUT24-ADCn_INPUT31
...	.	.....

23:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20:16	INPUT16TO23SEL	0x00	RW	**Inputs Chosen for ADCn_INPUT16-ADCn_INPUT23 as Referred in SCANMASK**

Mode	Value	Description
APORT0CH0TO7	0	Select APORT0's CH0-CH7 as ADCn_INPUT16-ADCn_INPUT23
APORT0CH8TO15	1	Select APORT0's CH8-CH15 as ADCn_INPUT16-ADCn_INPUT23
APORT1CH0TO7	4	Select APORT1's CH0-CH7 as ADCn_INPUT16-ADCn_INPUT23

Bit	Name	Reset	Access	Description
	APORT1CH8TO15	5		Select APORT1's CH8-CH15 as ADCn_INPUT16-ADCn_INPUT23
	APORT1CH16TO23	6		Select APORT1's CH16-CH23 as ADCn_INPUT16-ADCn_INPUT23
	APORT1CH24TO31	7		Select APORT1's CH24-CH31 as ADCn_INPUT16-ADCn_INPUT23
	APORT2CH0TO7	8		Select APORT2's CH0-CH7 as ADCn_INPUT16-ADCn_INPUT23
	...	.		.....
	APORT3CH0TO7	12		Select APORT3's CH0-CH7 as ADCn_INPUT16-ADCn_INPUT23
	...	.		.....
	APORT4CH0TO7	16		Select APORT4's CH0-CH7 as ADCn_INPUT16-ADCn_INPUT23
	...	.		.....
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:8	INPUT8TO15SEL	0x00	RW	**Inputs Chosen for ADCn_INPUT8-ADCn_INPUT15 as Referred in SCANMASK**

Mode	Value	Description
APORT0CH0TO7	0	Select APORT0's CH0-CH7 as ADCn_INPUT8-ADCn_INPUT15
APORT0CH8TO15	1	Select APORT0's CH8-CH15 as ADCn_INPUT8-ADCn_INPUT15
APORT1CH0TO7	4	Select APORT1's CH0-CH7 as ADCn_INPUT8-ADCn_INPUT15
APORT1CH8TO15	5	Select APORT1's CH8-CH15 as ADCn_INPUT8-ADCn_INPUT15
APORT1CH16TO23	6	Select APORT1's CH16-CH23 as ADCn_INPUT8-ADCn_INPUT15
APORT1CH24TO31	7	Select APORT1's CH24-CH31 as ADCn_INPUT8-ADCn_INPUT15
APORT2CH0TO7	8	Select APORT2's CH0-CH7 as ADCn_INPUT8-ADCn_INPUT15
...	.	.....
APORT3CH0TO7	12	Select APORT3's CH0-CH7 as ADCn_INPUT8-ADCn_INPUT15
...	.	.....
APORT4CH0TO7	16	Select APORT4's CH0-CH7 as ADCn_INPUT8-ADCn_INPUT15
...	.	.....

7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4:0	INPUT0TO7SEL	0x00	RW	**Inputs Chosen for ADCn_INPUT7-ADCn_INPUT0 as Referred in SCANMASK**

Mode	Value	Description
APORT0CH0TO7	0	Select APORT0's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
APORT0CH8TO15	1	Select APORT0's CH8-CH15 as ADCn_INPUT0-ADCn_INPUT7
APORT1CH0TO7	4	Select APORT1's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
APORT1CH8TO15	5	Select APORT1's CH8-CH15 as ADCn_INPUT0-ADCn_INPUT7
APORT1CH16TO23	6	Select APORT1's CH16-CH23 as ADCn_INPUT0-ADCn_INPUT7

Bit	Name	Reset	Access	Description
	APORT1CH24TO31	7		Select APORT1's CH24-CH31 as ADCn_INPUT0-ADCn_INPUT7
	APORT2CH0TO7	8		Select APORT2's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
	...	.		.....
	APORT3CH0TO7	12		Select APORT3's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
	...	.		.....
	APORT4CH0TO7	16		Select APORT4's CH0-CH7 as ADCn_INPUT0-ADCn_INPUT7
	...	.		.....

28.5.10 ADCn_SCANNEGSEL - Negative Input Select Register for Scan

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		0x3		0x2		0x1		0x3		0x2		0x1		0x0	
Access																	RW		RW		RW		RW		RW		RW		RW		RW	
Name																	INPUT15NEGSEL		INPUT13NEGSEL		INPUT11NEGSEL		INPUT9NEGSEL		INPUT6NEGSEL		INPUT4NEGSEL		INPUT2NEGSEL		INPUT0NEGSEL	

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*		
15:14	INPUT15NEGSEL	0x0	RW	**Negative Input Select Register for ADCn_INPUT15 in Differential Scan Mode**

Selects negative channel

Value	Mode	Description
0	INPUT8	Selects ADCn_INPUT8 as negative channel input
1	INPUT10	Selects ADCn_INPUT10 as negative channel input
2	INPUT12	Selects ADCn_INPUT12 as negative channel input
3	INPUT14	Selects ADCn_INPUT14 as negative channel input

Bit	Name	Reset	Access	Description
13:12	INPUT13NEGSEL	0x3	RW	**Negative Input Select Register for ADCn_INPUT13 in Differential Scan Mode**

Selects negative channel

Value	Mode	Description
0	INPUT8	Selects ADCn_INPUT8 as negative channel input
1	INPUT10	Selects ADCn_INPUT10 as negative channel input
2	INPUT12	Selects ADCn_INPUT12 as negative channel input
3	INPUT14	Selects ADCn_INPUT14 as negative channel input

Bit	Name	Reset	Access	Description
11:10	INPUT11NEGSEL	0x2	RW	**Negative Input Select Register for ADCn_INPUT11 in Differential Scan Mode**

Selects negative channel

Value	Mode	Description
0	INPUT8	Selects ADCn_INPUT8 as negative channel input
1	INPUT10	Selects ADCn_INPUT10 as negative channel input
2	INPUT12	Selects ADCn_INPUT12 as negative channel input
3	INPUT14	Selects ADCn_INPUT14 as negative channel input

Bit	Name	Reset	Access	Description
9:8	INPUT9NEGSEL	0x1	RW	**Negative Input Select Register for ADCn_INPUT9 in Differential Scan Mode**

Selects negative channel

Value	Mode	Description
0	INPUT8	Selects ADCn_INPUT8 as negative channel input
1	INPUT10	Selects ADCn_INPUT10 as negative channel input
2	INPUT12	Selects ADCn_INPUT12 as negative channel input
3	INPUT14	Selects ADCn_INPUT14 as negative channel input

Bit	Name	Reset	Access	Description
7:6	INPUT6NEGSEL	0x3	RW	**Negative Input Select Register for ADCn_INPUT1 in Differential Scan Mode**

Selects negative channel

Value	Mode	Description
0	INPUT1	Selects ADCn_INPUT1 as negative channel input
1	INPUT3	Selects ADCn_INPUT3 as negative channel input
2	INPUT5	Selects ADCn_INPUT5 as negative channel input
3	INPUT7	Selects ADCn_INPUT7 as negative channel input

Bit	Name	Reset	Access	Description
5:4	INPUT4NEGSEL	0x2	RW	**Negative Input Select Register for ADCn_INPUT4 in Differential Scan Mode**

Selects negative channel

Value	Mode	Description
0	INPUT1	Selects ADCn_INPUT1 as negative channel input
1	INPUT3	Selects ADCn_INPUT3 as negative channel input
2	INPUT5	Selects ADCn_INPUT5 as negative channel input
3	INPUT7	Selects ADCn_INPUT7 as negative channel input

Bit	Name	Reset	Access	Description
3:2	INPUT2NEGSEL	0x1	RW	**Negative Input Select Register for ADCn_INPUT2 in Differential Scan Mode**

Selects negative channel

Value	Mode	Description
0	INPUT1	Selects ADCn_INPUT1 as negative channel input
1	INPUT3	Selects ADCn_INPUT3 as negative channel input
2	INPUT5	Selects ADCn_INPUT5 as negative channel input
3	INPUT7	Selects ADCn_INPUT7 as negative channel input

Bit	Name	Reset	Access	Description
1:0	INPUT0NEGSEL	0x0	RW	**Negative Input Select Register for ADCn_INPUT0 in Differential Scan Mode**

Selects negative channel

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	INPUT1		Selects ADCn_INPUT1 as negative channel input
	1	INPUT3		Selects ADCn_INPUT3 as negative channel input
	2	INPUT5		Selects ADCn_INPUT5 as negative channel input
	3	INPUT7		Selects ADCn_INPUT7 as negative channel input

28.5.11 ADCn_CMPTHR - Compare Threshold 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	0x0000																0x0000															
Access	RW																RW															
Name	ADGT																ADLT															

Bit	Name	Reset	Access	Description
31:16	ADGT	0x0000	RW	**Greater Than Compare Threshold**
	Compare threshold value for greater-than comparison. Must match the conversion data representation chosen.			
15:0	ADLT	0x0000	RW	**Less Than Compare Threshold**
	Compare threshold value for less-than comparison. Must match the conversion data representation chosen.			

28.5.12 ADCn_BIASPROG - Bias Programming Register for Various Analog Blocks Used in ADC Operation

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																0				0									0x0			
Access																RW				RW									RW			
Name																GPBIASACC				VFAULTCLR									ADCBIASPROG			

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	GPBIASACC	0	RW	**Accuracy Setting for the System Bias During ADC Operation**

Select bias accuracy mode for ADC operation. For devices with multiple ADCs, the bias will use the high accuracy setting unless all ADC instances configure GPBIASACC to LOWACC.

Value	Mode	Description
0	HIGHACC	High accuracy setting. Use when configured for an internal VBGR reference source.
1	LOWACC	Low accuracy setting. Can be used for all references other than VBGR to conserve energy.

15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	VFAULTCLR	0	RW	**Clear VREFOF Flag**

Use this bit to request clearing of the VREFOF flag. If VREFOF irq is enabled and is triggered, the user must set this bit in the ISR to clear VREFOF. The user needs to reset this bit to enable VREFOF to trigger further IRQs upon VREF overflow conditions.

11:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	ADCBIASPROG	0x0	RW	**Bias Programming Value of Analog ADC Block**

These bits are used to adjust the bias current in ADC analog block.

Value	Mode	Description
0	NORMAL	Normal power (use for 1Msps operation)
4	SCALE2	Scaling bias to 1/2
8	SCALE4	Scaling bias to 1/4
12	SCALE8	Scaling bias to 1/8
14	SCALE16	Scaling bias to 1/16
15	SCALE32	Scaling bias to 1/32

28.5.13 ADCn_CAL - Calibration 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	0	0x40							0x7				0x8				0	0x40							0x7				0x8			
Access	RW	RW							RW				RW				RW	RW							RW				RW			
Name	CALEN	SCANGAIN							SCANOFFSETINV				SCANOFFSET				OFFSETINVMODE	SINGLEGAIN							SINGLEOFFSETINV				SINGLEOFFSET			

Bit	Name	Reset	Access	Description
31	CALEN	0	RW	**Calibration Mode is Enabled**
	When enabled, the adc performs conversion and sends raw data to the ADC fifos. This can also be used to debug the adc data conversion			
30:24	SCANGAIN	0x40	RW	**Scan Mode Gain Calibration Value**
	This register contains the gain calibration value used with scan conversions. This field is set to the production gain calibration value for the 1V25 internal reference during reset, hence the reset value might differ from device to device. The field is unsigned. Higher values lead to higher ADC results.			
23:20	SCANOFFSETINV	0x7	RW	**Scan Mode Offset Calibration Value for Negative Single-ended Mode**
	This register contains the offset calibration value used with scan conversions for negative single-ended mode. This field is set to the production offset calibration value for the 1V25 internal reference during reset, hence the reset value might differ from device to device. The field is encoded as a signed 2's complement number. Higher values lead to lower ADC results.			
19:16	SCANOFFSET	0x8	RW	**Scan Mode Offset Calibration Value for Differential or Positive Single-ended Mode**
	This register contains the offset calibration value used with scan conversions for differential or positive single-ended mode. This field is set to the production offset calibration value for the 1V25 internal reference during reset, hence the reset value might differ from device to device. The field is encoded as a signed 2's complement number. Higher values lead to lower ADC results.			
15	OFFSETINVMODE	0	RW	**Negative Single-ended Offset Calibration is Enabled**
	When enabled, along with CALEN bit, the ADC performs negative singled ended conversion. When not enabled, if CALEN is set, DIFF bit of SINGLECTRL register decides whether to do positive single-ended or differential conversion.			
14:8	SINGLEGAIN	0x40	RW	**Single Mode Gain Calibration Value**
	This register contains the gain calibration value used with single conversions. This field is set to the production gain calibration value for the 1V25 internal reference during reset, hence the reset value might differ from device to device. The field is unsigned. Higher values lead to higher ADC results.			
7:4	SINGLEOFFSETINV	0x7	RW	**Single Mode Offset Calibration Value for Negative Single-ended Mode**
	This register contains the offset calibration value used with single conversions for negative single-ended mode. This field is set to the production offset calibration value for the 1V25 internal reference during reset, hence the reset value might differ from device to device. The field is encoded as a signed 2's complement number. Higher values lead to lower ADC results.			

Bit	Name	Reset	Access	Description
3:0	SINGLEOFFSET	0x8	RW	**Single Mode Offset Calibration Value for Differential or Positive Single-ended Mode** This register contains the offset calibration value used with single conversions for differential or positive single-ended mode. This field is set to the production offset calibration value for the 1V25 internal reference during reset, hence the reset value might differ from device to device. The field is encoded as a signed 2's complement number. Higher values lead to lower ADC results.

28.5.14 ADCn_IF - Interrupt Flag 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			0	0	0	0	0	0							0	0					0	0	0	0							0	0
Access			R	R	R	R	R	R							R	R					R	R	R	R							R	R
Name			EM23ERR	PRSTIMEDERR	SCANPEND	SCANEXTPEND	PROGERR	VREFOV							SCANCMP	SINGLECMP					SCANUF	SINGLEUF	SCANOF	SINGLEOF							SCAN	SINGLE

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	EM23ERR	0	R	**EM23 Entry Error Flag**
	Indicates that an incorrect clock was selected as ADC_CLK when going into EM23 resulting in an incorrect conversion.			
28	PRSTIMEDERR	0	R	**PRS Timed Mode Error Flag**
	Indicates that in PRS timed mode, a PRS negative edge arrived before the AT event and it was ignored.			
27	SCANPEND	0	R	**Scan Trigger Pending Flag**
	Indicates that an external scan (e.g., LESENSE triggered) is running and PRS/software triggered scan has gone pending.			
26	SCANEXTPEND	0	R	**External Scan Trigger Pending Flag**
	Indicates that a PRS/software triggered scan is running and the external scan (e.g., LESENSE triggered) has gone pending.			
25	PROGERR	0	R	**Programming Error Interrupt Flag**
	Indicates that a programming error has occurred. Read the STATUS register for cause.			
24	VREFOV	0	R	**VREF Over Voltage Interrupt Flag**
	Indicates that attenuated vref is greater than 1.3V when this bit is set. The ADC stops converting and disconnects the reference when this happens to protect the internal low-voltage circuits.			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	SCANCMP	0	R	**Scan Result Compare Match Interrupt Flag**
	Indicates scan result compare matched the window conditions when this bit is set.			
16	SINGLECMP	0	R	**Single Result Compare Match Interrupt Flag**
	Indicates single result compare matched the window conditions when this bit is set.			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	SCANUF	0	R	**Scan FIFO Underflow Interrupt Flag**
	Indicates scan result FIFO underflow when this bit is set. An underflow occurs if the FIFO is read and there is no data available.			
10	SINGLEUF	0	R	**Single FIFO Underflow Interrupt Flag**
	Indicates single result FIFO underflow when this bit is set. An underflow occurs if the FIFO is read and there is no data available.			

Bit	Name	Reset	Access	Description
9	SCANOF	0	R	**Scan FIFO Overflow Interrupt Flag**
	Indicates scan result FIFO overflow when this bit is set. An overflow occurs if there is not room in the FIFO to store a new result.			
8	SINGLEOF	0	R	**Single FIFO Overflow Interrupt Flag**
	Indicates single result FIFO overflow when this bit is set. An overflow occurs if there is not room in the FIFO to store a new result.			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	SCAN	0	R	**Scan Conversion Complete Interrupt Flag**
	Indicates (DVL+1) number of scan channel results are available in the Scan FIFO.			
0	SINGLE	0	R	**Single Conversion Complete Interrupt Flag**
	Indicates (DVL+1) number of single channel results are available in the Single FIFO.			

28.5.15 ADCn_IFS - Interrupt Flag Set 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			0	0	0	0	0	0							0	0					0	0	0	0								
Access			W1	W1	W1	W1	W1	W1							W1	W1					W1	W1	W1	W1								
Name			EM23ERR	PRSTIMEDERR	SCANPEND	SCANEXTPEND	PROGERR	VREFOV							SCANCMP	SINGLECMP					SCANUF	SINGLEUF	SCANOF	SINGLEOF								

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	EM23ERR	0	W1	**Set EM23ERR Interrupt Flag**
	Write 1 to set the EM23ERR interrupt flag			
28	PRSTIMEDERR	0	W1	**Set PRSTIMEDERR Interrupt Flag**
	Write 1 to set the PRSTIMEDERR interrupt flag			
27	SCANPEND	0	W1	**Set SCANPEND Interrupt Flag**
	Write 1 to set the SCANPEND interrupt flag			
26	SCANEXTPEND	0	W1	**Set SCANEXTPEND Interrupt Flag**
	Write 1 to set the SCANEXTPEND interrupt flag			
25	PROGERR	0	W1	**Set PROGERR Interrupt Flag**
	Write 1 to set the PROGERR interrupt flag			
24	VREFOV	0	W1	**Set VREFOV Interrupt Flag**
	Write 1 to set the VREFOV interrupt flag			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	SCANCMP	0	W1	**Set SCANCMP Interrupt Flag**
	Write 1 to set the SCANCMP interrupt flag			
16	SINGLECMP	0	W1	**Set SINGLECMP Interrupt Flag**
	Write 1 to set the SINGLECMP interrupt flag			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	SCANUF	0	W1	**Set SCANUF Interrupt Flag**
	Write 1 to set the SCANUF interrupt flag			
10	SINGLEUF	0	W1	**Set SINGLEUF Interrupt Flag**
	Write 1 to set the SINGLEUF interrupt flag			
9	SCANOF	0	W1	**Set SCANOF Interrupt Flag**
	Write 1 to set the SCANOF interrupt flag			

Bit	Name	Reset	Access	Description
8	SINGLEOF	0	W1	**Set SINGLEOF Interrupt Flag**
	Write 1 to set the SINGLEOF interrupt flag			
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

28.5.16 ADCn_IFC - Interrupt Flag Clear 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			0	0	0	0	0	0							0	0					0	0	0	0								
Access			(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1							(R)W1	(R)W1					(R)W1	(R)W1	(R)W1	(R)W1								
Name			EM23ERR	PRSTIMEDERR	SCANPEND	SCANEXTPEND	PROGERR	VREFOV							SCANCMP	SINGLECMP					SCANUF	SINGLEUF	SCANOF	SINGLEOF								

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	EM23ERR	0	(R)W1	**Clear EM23ERR Interrupt Flag**
	Write 1 to clear the EM23ERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
28	PRSTIMEDERR	0	(R)W1	**Clear PRSTIMEDERR Interrupt Flag**
	Write 1 to clear the PRSTIMEDERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
27	SCANPEND	0	(R)W1	**Clear SCANPEND Interrupt Flag**
	Write 1 to clear the SCANPEND interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
26	SCANEXTPEND	0	(R)W1	**Clear SCANEXTPEND Interrupt Flag**
	Write 1 to clear the SCANEXTPEND interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
25	PROGERR	0	(R)W1	**Clear PROGERR Interrupt Flag**
	Write 1 to clear the PROGERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
24	VREFOV	0	(R)W1	**Clear VREFOV Interrupt Flag**
	Write 1 to clear the VREFOV interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	SCANCMP	0	(R)W1	**Clear SCANCMP Interrupt Flag**
	Write 1 to clear the SCANCMP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
16	SINGLECMP	0	(R)W1	**Clear SINGLECMP Interrupt Flag**
	Write 1 to clear the SINGLECMP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
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	SCANUF	0	(R)W1	**Clear SCANUF Interrupt Flag**
	Write 1 to clear the SCANUF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
10	SINGLEUF	0	(R)W1	**Clear SINGLEUF Interrupt Flag**
	Write 1 to clear the SINGLEUF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	SCANOF	0	(R)W1	**Clear SCANOF Interrupt Flag**
	Write 1 to clear the SCANOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	SINGLEOF	0	(R)W1	**Clear SINGLEOF Interrupt Flag**
	Write 1 to clear the SINGLEOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

28.5.17 ADCn_IEN - Interrupt Enable 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			0	0	0	0	0	0							0	0					0	0	0	0							0	0
Access			RW	RW	RW	RW	RW	RW							RW	RW					RW	RW	RW	RW							RW	RW
Name			EM23ERR	PRSTIMEDERR	SCANPEND	SCANEXTPEND	PROGERR	VREFOV							SCANCMP	SINGLECMP					SCANUF	SINGLEUF	SCANOF	SINGLEOF							SCAN	SINGLE

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	EM23ERR	0	RW	**EM23ERR Interrupt Enable**
	Enable/disable the EM23ERR interrupt			
28	PRSTIMEDERR	0	RW	**PRSTIMEDERR Interrupt Enable**
	Enable/disable the PRSTIMEDERR interrupt			
27	SCANPEND	0	RW	**SCANPEND Interrupt Enable**
	Enable/disable the SCANPEND interrupt			
26	SCANEXTPEND	0	RW	**SCANEXTPEND Interrupt Enable**
	Enable/disable the SCANEXTPEND interrupt			
25	PROGERR	0	RW	**PROGERR Interrupt Enable**
	Enable/disable the PROGERR interrupt			
24	VREFOV	0	RW	**VREFOV Interrupt Enable**
	Enable/disable the VREFOV interrupt			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	SCANCMP	0	RW	**SCANCMP Interrupt Enable**
	Enable/disable the SCANCMP interrupt			
16	SINGLECMP	0	RW	**SINGLECMP Interrupt Enable**
	Enable/disable the SINGLECMP interrupt			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	SCANUF	0	RW	**SCANUF Interrupt Enable**
	Enable/disable the SCANUF interrupt			
10	SINGLEUF	0	RW	**SINGLEUF Interrupt Enable**
	Enable/disable the SINGLEUF interrupt			
9	SCANOF	0	RW	**SCANOF Interrupt Enable**
	Enable/disable the SCANOF interrupt			

Bit	Name	Reset	Access	Description
8	SINGLEOF	0	RW	**SINGLEOF Interrupt Enable**
	Enable/disable the SINGLEOF interrupt			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	SCAN	0	RW	**SCAN Interrupt Enable**
	Enable/disable the SCAN interrupt			
0	SINGLE	0	RW	**SINGLE Interrupt Enable**
	Enable/disable the SINGLE interrupt			

28.5.18 ADCn_SINGLEDATA - Single Conversion Result Data (Actionable Reads)

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	0x00000000																															
Access	R																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	R	**Single Conversion Result Data**
	This register holds the results from the last single channel mode conversion. Reading this field pops one entry from the SINGLE FIFO.			

28.5.19 ADCn_SCANDATA - Scan Conversion Result Data (Actionable Reads)

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	0x00000000																															
Access	R																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	R	**Scan Conversion Result Data**
	The register holds the results from the last scan mode conversion. Reading this field pops one entry from the SCAN FIFO.			

28.5.20 ADCn_SINGLEDATAP - Single Conversion Result Data Peek 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	0x00000000																															
Access	R																															
Name	DATAP																															

Bit	Name	Reset	Access	Description
31:0	DATAP	0x00000000	R	**Single Conversion Result Data Peek**
	The register holds the results from the last single channel mode conversion. Reading this field will not pop an entry from the SINGLE FIFO.			

28.5.21 ADCn_SCANDATAP - Scan Sequence Result Data Peek 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	0x00000000																															
Access	R																															
Name	DATAP																															

Bit	Name	Reset	Access	Description
31:0	DATAP	0x00000000	R	**Scan Conversion Result Data Peek**
	The register holds the results from the last scan mode conversion. Reading this field will not pop an entry from the SCAN FIFO.			

28.5.22 ADCn_SCANDATAX - Scan Sequence Result Data + Data Source Register (Actionable Reads)

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														0x00			0x0000															
Access														R			R															
Name														SCANINPUTID			DATA															

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*		
20:16	SCANINPUTID	0x00	R	**Scan Conversion Input ID**
	Indicates from which input the results in SCANDATA originated. Reading this field pops one entry from the SCAN FIFO.			
15:0	DATA	0x0000	R	**Scan Conversion Result Data**
	Holds the results from the last scan conversion. Reading this field pops one entry from the SCAN FIFO.			

28.5.23 ADCn_SCANDATAXP - Scan Sequence Result Data + Data Source Peek Register

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														0x00			0x0000															
Access														R			R															
Name														SCANINPUTIDPEEK			DATAP															

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*		
20:16	SCANINPUTIDPEEK	0x00	R	**Scan Conversion Data Source Peek**
	Indicates from which input channel the results in SCANDATA originated. Reading this field does not pop any entry from the SCAN FIFO.			
15:0	DATAP	0x0000	R	**Scan Conversion Result Data Peek**
	The register holds the results from the last scan conversion. Reading this field does not pop any entry from the SCAN FIFO.			

28.5.24 ADCn_APORTREQ - APORT Request Status Register

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																							0	0	0	0	0	0	0	0	0	0
Access																							R	R	R	R	R	R	R	R	R	R
Name																							APORT4YREQ	APORT4XREQ	APORT3YREQ	APORT3XREQ	APORT2YREQ	APORT2XREQ	APORT1YREQ	APORT1XREQ	APORT0YREQ	APORT0XREQ

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	APORT4YREQ	0	R	**1 If the Bus Connected to APORT4Y is Requested**
	Reports if the bus connected to APORT4Y is being requested from the APORT			
8	APORT4XREQ	0	R	**1 If the Bus Connected to APORT4X is Requested**
	Reports if the bus connected to APORT4X is being requested from the APORT			
7	APORT3YREQ	0	R	**1 If the Bus Connected to APORT3Y is Requested**
	Reports if the bus connected to APORT3Y is being requested from the APORT			
6	APORT3XREQ	0	R	**1 If the Bus Connected to APORT3X is Requested**
	Reports if the bus connected to APORT3X is being requested from the APORT			
5	APORT2YREQ	0	R	**1 If the Bus Connected to APORT2Y is Requested**
	Reports if the bus connected to APORT2Y is being requested from the APORT			
4	APORT2XREQ	0	R	**1 If the Bus Connected to APORT2X is Requested**
	Reports if the bus connected to APORT2X is being requested from the APORT			
3	APORT1YREQ	0	R	**1 If the Bus Connected to APORT1Y is Requested**
	Reports if the bus connected to APORT1Y is being requested from the APORT			
2	APORT1XREQ	0	R	**1 If the Bus Connected to APORT1X is Requested**
	Reports if the bus connected to APORT1X is being requested from the APORT			
1	APORT0YREQ	0	R	**1 If the Bus Connected to APORT0Y is Requested**
	Reports if the bus connected to APORT0Y is being requested from the APORT			
0	APORT0XREQ	0	R	**1 If the Bus Connected to APORT0X is Requested**
	Reports if the bus connected to APORT0X is being requested from the APORT			

28.5.25 ADCn_APORTCONFLICT - APORT Conflict Status 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																							0	0	0	0	0	0	0	0	0	0
Access																							R	R	R	R	R	R	R	R	R	R
Name																							APORT4YCONFLICT	APORT4XCONFLICT	APORT3YCONFLICT	APORT3XCONFLICT	APORT2YCONFLICT	APORT2XCONFLICT	APORT1YCONFLICT	APORT1XCONFLICT	APORT0YCONFLICT	APORT0XCONFLICT

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	APORT4YCONFLICT	0	R	**1 If the Bus Connected to APORT4Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT4Y is is also being requested by another peripheral
8	APORT4XCONFLICT	0	R	**1 If the Bus Connected to APORT4X is in Conflict With Another Peripheral** Reports if the bus connected to APORT4X is is also being requested by another peripheral
7	APORT3YCONFLICT	0	R	**1 If the Bus Connected to APORT3Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT3Y is is also being requested by another peripheral
6	APORT3XCONFLICT	0	R	**1 If the Bus Connected to APORT3X is in Conflict With Another Peripheral** Reports if the bus connected to APORT3X is is also being requested by another peripheral
5	APORT2YCONFLICT	0	R	**1 If the Bus Connected to APORT2Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT2Y is is also being requested by another peripheral
4	APORT2XCONFLICT	0	R	**1 If the Bus Connected to APORT2X is in Conflict With Another Peripheral** Reports if the bus connected to APORT2X is is also being requested by another peripheral
3	APORT1YCONFLICT	0	R	**1 If the Bus Connected to APORT1Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT1Y is is also being requested by another peripheral
2	APORT1XCONFLICT	0	R	**1 If the Bus Connected to APORT1X is in Conflict With Another Peripheral** Reports if the bus connected to APORT1X is is also being requested by another peripheral
1	APORT0YCONFLICT	0	R	**1 If the Bus Connected to APORT0Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT0Y is is also being requested by another peripheral

Bit	Name	Reset	Access	Description
0	APORT0XCONFLICT	0	R	**1 If the Bus Connected to APORT0X is in Conflict With Another Peripheral** Reports if the bus connected to APORT0X is is also being requested by another peripheral

28.5.26 ADCn_SINGLEFIFOCOUNT - Single FIFO Count 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																															R	
Name																															SINGLEDC	

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	SINGLEDC	0x0	R	**Single Data Count** Number of unread data available in Single FIFO.

28.5.27 ADCn_SCANFIFOCOUNT - Scan FIFO Count Register

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																															SCANDC	

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	SCANDC	0x0	R	**Scan Data Count** Number of unread data available in Scan FIFO.

28.5.28 ADCn_SINGLEFIFOCLEAR - Single FIFO Clear Register

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																																0
Access																																W1
Name																																SINGLEFIFOCLEAR

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	SINGLEFIFOCLEAR	0	W1	**Clear Single FIFO Content**
	Write a 1 to clear Single FIFO content.			

28.5.29 ADCn_SCANFIFOCLEAR - Scan FIFO Clear 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																																0
Access																																W1
Name																																SCANFIFOCLEAR

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	SCANFIFOCLEAR	0	W1	**Clear Scan FIFO Content**
	Write a 1 to clear Scan FIFO content.			

28.5.30 ADCn_APORTMASTERDIS - APORT Bus Master Disable Register

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																							0	0	0	0	0	0	0	0		
Access																							RW	RW	RW	RW	RW	RW	RW	RW		
Name																							APORT4YMASTERDIS	APORT4XMASTERDIS	APORT3YMASTERDIS	APORT3XMASTERDIS	APORT2YMASTERDIS	APORT2XMASTERDIS	APORT1YMASTERDIS	APORT1XMASTERDIS		

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	APORT4YMASTERDIS	0	RW	**APORT4Y Master Disable**

Determines if the ADC will request this APORT bus (if selected by POSSEL or NEGSEL or SCANINPUTSEL). When 1, ADC only passively monitors the APORT bus and the selection of the channel for the selected bus is ignored. The channel selection is done by the device that masters the APORT bus. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously.

Value	Description
0	APORT mastering enabled
1	APORT mastering disabled

8	APORT4XMASTERDIS	0	RW	**APORT4X Master Disable**

Determines if the ADC will request this APORT bus (if selected by POSSEL or NEGSEL or SCANINPUTSEL). When 1, ADC only passively monitors the APORT bus and the selection of the channel for the selected bus is ignored. The channel selection is done by the device that masters the APORT bus. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously.

Value	Description
0	APORT mastering enabled
1	APORT mastering disabled

7	APORT3YMASTERDIS	0	RW	**APORT3Y Master Disable**

Determines if the ADC will request this APORT bus (if selected by POSSEL or NEGSEL or SCANINPUTSEL). When 1, ADC only passively monitors the APORT bus and the selection of the channel for the selected bus is ignored. The channel selection is done by the device that masters the APORT bus. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously.

Value	Description
0	APORT mastering enabled
1	APORT mastering disabled

Bit	Name	Reset	Access	Description
6	APORT3XMASTER-DIS	0	RW	**APORT3X Master Disable**

Determines if the ADC will request this APORT bus (if selected by POSSEL or NEGSEL or SCANINPUTSEL). When 1, ADC only passively monitors the APORT bus and the selection of the channel for the selected bus is ignored. The channel selection is done by the device that masters the APORT bus. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously.

Value	Description
0	APORT mastering enabled
1	APORT mastering disabled

Bit	Name	Reset	Access	Description
5	APORT2YMASTER-DIS	0	RW	**APORT2Y Master Disable**

Determines if the ADC will request this APORT bus (if selected by POSSEL or NEGSEL or SCANINPUTSEL). When 1, ADC only passively monitors the APORT bus and the selection of the channel for the selected bus is ignored. The channel selection is done by the device that masters the APORT bus. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously.

Value	Description
0	APORT mastering enabled
1	APORT mastering disabled

Bit	Name	Reset	Access	Description
4	APORT2XMASTER-DIS	0	RW	**APORT2X Master Disable**

Determines if the ADC will request this APORT bus (if selected by POSSEL or NEGSEL or SCANINPUTSEL). When 1, ADC only passively monitors the APORT bus and the selection of the channel for the selected bus is ignored. The channel selection is done by the device that masters the APORT bus. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously.

Value	Description
0	APORT mastering enabled
1	APORT mastering disabled

Bit	Name	Reset	Access	Description
3	APORT1YMASTER-DIS	0	RW	**APORT1Y Master Disable**

Determines if the ADC will request this APORT bus (if selected by POSSEL or NEGSEL or SCANINPUTSEL). When 1, ADC only passively monitors the APORT bus and the selection of the channel for the selected bus is ignored. The channel selection is done by the device that masters the APORT bus. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously.

Value	Description
0	APORT mastering enabled
1	APORT mastering disabled

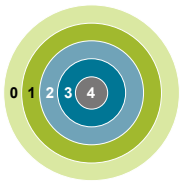
Bit	Name	Reset	Access	Description
2	APORT1XMASTER-DIS	0	RW	**APORT1X Master Disable**

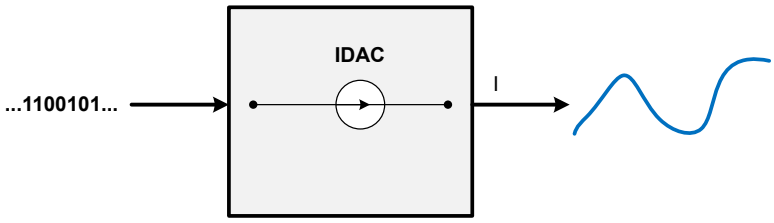
Determines if the ADC will request this APORT bus (if selected by POSSEL or NEGSEL or SCANINPUTSEL). When 1, ADC only passively monitors the APORT bus and the selection of the channel for the selected bus is ignored. The channel selection is done by the device that masters the APORT bus. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously.

Value	Description

Bit	Name	Reset	Access	Description
	0			APORT mastering enabled
	1			APORT mastering disabled
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

29. IDAC - Current Digital to Analog Converter





Quick Facts

What?

The IDAC can sink or source a configurable constant current.

Why?

The IDAC can be used to bias external circuits or (in conjunction with the ADC) measure capacitance by injecting a controlled current into a component.

How?

In addition to providing a constant current, the IDAC can be switched on and off with a PRS signal all the way down to EM3.

29.1 Introduction

The current digital to analog converter (IDAC) can source or sink a configurable constant current from APORT and/or main pad (OUTPAD). The current is configurable with several ranges of various step sizes.

29.2 Features

- Can source and sink current
- Programmable constant output current
  - Selectable current range between 0.05 µA and 64 µA
  - Each range is linearly programmable in 32 steps
  - Support for current calibration
- Support for manual and PRS triggered output enable
- Available in EM0-EM3

29.3 Functional Description

An overview of the IDAC module is shown in Figure 29.1 IDAC Overview on page 1086. The IDAC is designed to source or sink a programmable current which can be controlled by setting the range and the step in the RANGESEL and STEPSEL bitfields in IDAC_CURRPROG register. The IDAC output enable can be controlled by software or PRS. The IDAC is enabled by setting EN in IDAC_CTRL.

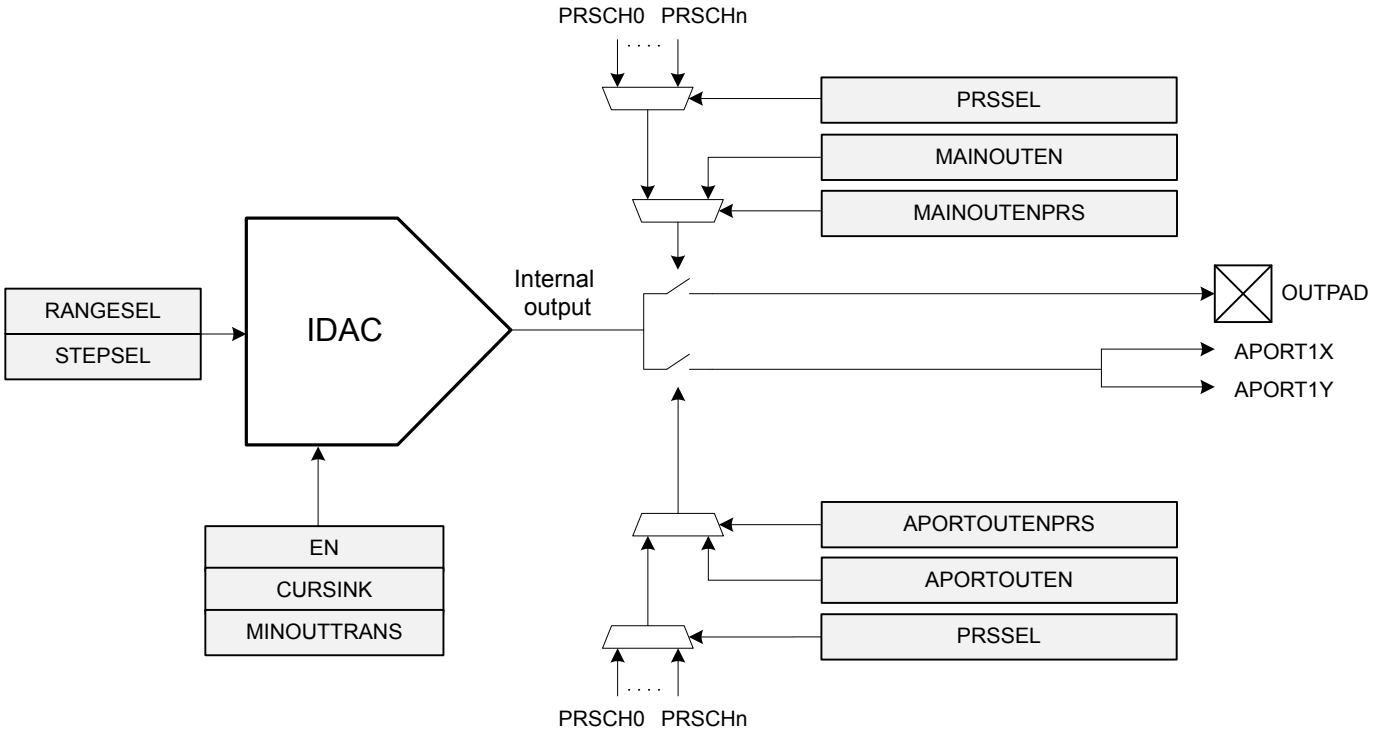


Figure 29.1. IDAC Overview

29.3.1 Current Programming

The four different current ranges can be selected by configuring the RANGESEL bitfield in IDAC_CURRPROG. The current output in each range is linearly programmable in 32 steps, and is controlled by the STEPSEL bitfield in IDAC_CURRPROG. These current ranges and their step sizes are shown in Table 29.1 Range Selection on page 1086.

Table 29.1. Range Selection

Range Select	Range Value [µA]	Step Size [nA]	Step Counts
0	0.05 - 1.6	50	32
1	1.6 - 4.7	100	32
2	0.5 - 16	500	32
3	2 - 64	2000	32

29.3.2 IDAC Enable and Warm-up

The IDAC is enabled by setting the EN bit in IDAC_CTRL. When this bit is set, the IDAC must stabilize before its output current is stable.

It is important to wait until the IDAC is warmed up, or until any current programming is complete and the output current is stabilized, before entering EM1, EM2, or EM3.

29.3.3 Output Control

The IDAC output enable is controlled separately for APORT and main pad. If APORTOUTENPRS/MAINOUTENPRS is set, output enable is controlled by PRS, else it is controlled by software via APORTOUTEN/MAINOUTEN.

29.3.4 APORT Configuration

The IDAC APORT outputs can be routed to pins through the APORT system. Note that the IDAC has only two local APORT interfaces APORT1X and APORT1Y, which are connected to the APORT BUSCX and BUSCY, respectively. The pins are selected by requesting an APORT channel in APORTOUTSEL in IDAC_CTRL. For mapping between APORT channel and physical pin, please refer to data sheet. The IDAC can be in either master or slave mode when connecting to the APORT. By default the IDAC is in master mode. To enable slave mode, set APORTMASTERDIS in IDAC_CTRL. As IDAC is only capable of driving an APORT channel, only master mode is meaningful for IDAC. If the IDAC is in master mode, and another module is currently driving the requested channel, the APORTCONFLICT bitfield in IDAC_STATUS will be set together with APORT1XCONFLICT or APORT1YCONFLICT in IDAC_APORTCONFLICT. The APORTCONFLICT can also be configured to trigger an interrupt, see 29.3.5 Interrupts for details.

29.3.5 Interrupts

The APORTCONFLICT interrupt flag in the IDAC_IF register indicates that a conflict has occurred when requesting a channel from the APORT. The APORTCONFLICT interrupt can be enabled by setting the APORTCONFLICT bit in IDAC_IEN, or cleared by setting the APORTCONFLICT bit in IDAC_IFC.

29.3.6 Minimizing Output Transition

If the internal output of the IDAC differs from the voltage at the output pin, enabling the output can cause an unwanted transition. To minimize this transition, it is possible to charge or discharge the internal output node before enabling the output to the pin. Setting MINOUTTRANS in IDAC_CTRL when the IDAC is sourcing current connects the internal node to GND. Alternatively, setting MINOUTTRANS when the IDAC is sinking current connects the internal output node to VDD. Setting APORTOUTEN/MAINOUTEN when MINOUTTRANS is set will halt the charge/discharge until either APORTOUTEN/MAINOUTEN is cleared or MINOUTTRANS is cleared.

29.3.7 Duty Cycle Configuration

The references for the IDAC can be duty-cycled, meaning that it can source current at very low overhead current consumption at the cost of response time and accuracy. By default duty-cycling is enabled in EM2 and EM3 and disabled in EM0 and EM1. Setting EM2DUTYCYCLEDIS in IDAC_DUTYCONFIG will disable duty cycling in EM2 and EM3. Note that sinking current can not be done with duty-cycled references so measures needs to be taken to always disable duty-cycling while sinking current.

29.3.8 Calibration

The IDAC can be calibrated to accurately compensate for process, supply voltage and temperature variations. During the production test, the middle step of each range is calibrated at room temperature. The TUNING bitfield in the IDAC_CAL register can be used to do further calibration of each step with an external resistor connected to IDAC_OUT. The calibrated tuning value for each band can be read from the Device Information (DI) page.

29.3.9 PRS Triggered Charge Injection

The amount of charge sourced or sunk by the IDAC can be controlled by the PRS (e.g., using a timer as producer) via the output switch. Figure 29.2 IDAC Charge Injection Example on page 1088 shows a case where the IDAC is configured to periodically supply charge using the PRS. The amount of charge injected is proportional to the the period the IDAC is on. The total charge injected is the current multiplied by the time the output switch is enabled.

The PRS system is enabled by setting APORTOUTENPRS/MAINOUTENPRS in IDAC_CTRL, and the PRS channel is selected by PRSSEL in IDAC_CTRL. To generate the periodic control signal, the TIMER module can be used, by configuring for example a CC channel to compare match with a configurable level.

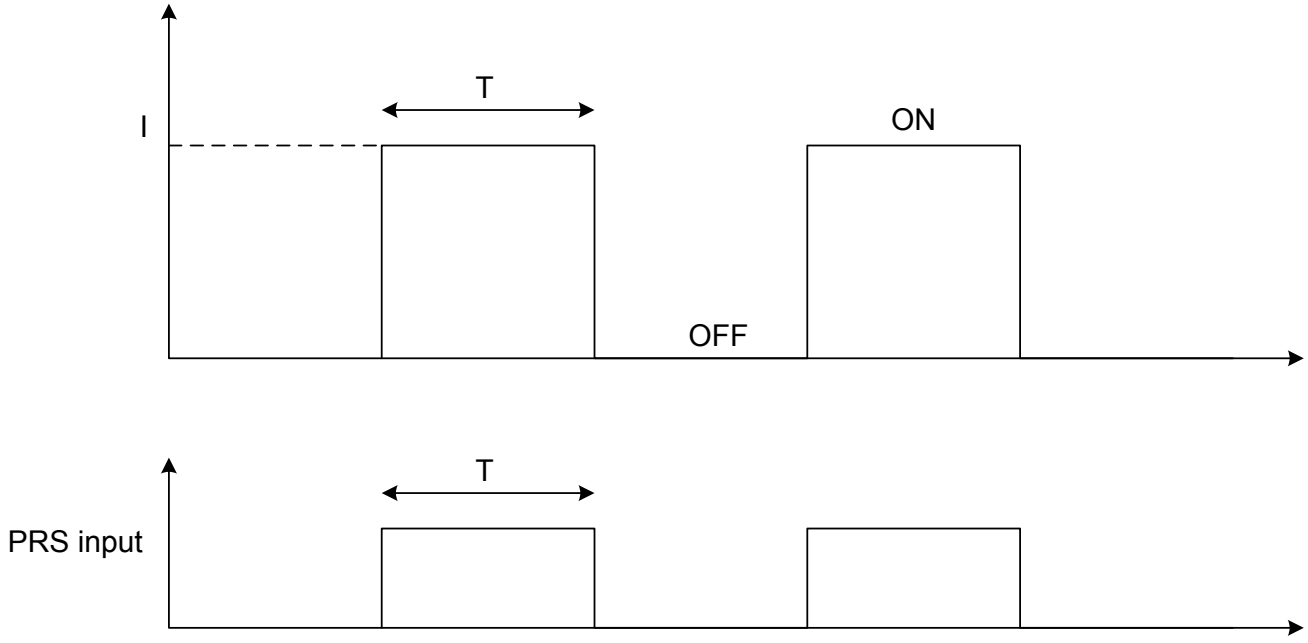


Figure 29.2. IDAC Charge Injection Example

29.4 Register Map

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

Offset	Name	Type	Description
0x000	IDAC_CTRL	RW	Control Register
0x004	IDAC_CURPROG	RW	Current Programming Register
0x00C	IDAC_DUTYCONFIG	RW	Duty Cycle Configuration Register
0x018	IDAC_STATUS	R	Status Register
0x020	IDAC_IF	R	Interrupt Flag Register
0x024	IDAC_IFS	W1	Interrupt Flag Set Register
0x028	IDAC_IFC	(R)W1	Interrupt Flag Clear Register
0x02C	IDAC_IEN	RW	Interrupt Enable Register
0x034	IDAC_APORTREQ	R	APORT Request Status Register
0x038	IDAC_APORTCONFLICT	R	APORT Request Status Register

29.5 Register Description

29.5.1 IDAC_CTRL - 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				0	0		0		0	0	0	0x00								0	0	0	0
Access									RW				RW	RW		RW		RW	RW	RW	RW								RW	RW	RW	RW
Name									PRSSEL				MAINOUTENPRS	MAINOUTEN		APORTOUTENPRS		APORTMASTERDIS	EM2DELAY	PWRSEL	APORTOUTSEL								APORTOUTEN	MINOUTTRANS	CURSINK	EN

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*		
23:20	PRSSEL	0x0	RW	**IDAC Output Enable PRS Channel Select**

Selects which PRS channel to use to enable output.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected.
1	PRSCH1	PRS Channel 1 selected.
2	PRSCH2	PRS Channel 2 selected.
3	PRSCH3	PRS Channel 3 selected.
4	PRSCH4	PRS Channel 4 selected.
5	PRSCH5	PRS Channel 5 selected.
6	PRSCH6	PRS Channel 6 selected.
7	PRSCH7	PRS Channel 7 selected.
8	PRSCH8	PRS Channel 8 selected.
9	PRSCH9	PRS Channel 9 selected.
10	PRSCH10	PRS Channel 10 selected.
11	PRSCH11	PRS Channel 11 selected.
12	PRSCH12	PRS Channel 12 selected.
13	PRSCH13	PRS Channel 13 selected.
14	PRSCH14	PRS Channel 14 selected.
15	PRSCH15	PRS Channel 15 selected.

19	MAINOUTENPRS	0	RW	**PRS Controlled Main Pad Output Enable**

Enable PRS Control of IDAC Main Pad output enable.

Value	Description

Bit	Name	Reset	Access	Description
	0			Main pad output enable controlled by IDAC_MAINOUTEN.
	1			Main pad output enable controlled by PRS input selected by PRSSEL.
18	MAINOUTEN	0	RW	**Output Enable**
	Set to enable IDAC main output to pad if MAINOUTENPRS is not set.			
17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	APORTOUTENPRS	0	RW	**PRS Controlled APORT Output Enable**
	Enable PRS Control of the IDAC APORT output enable.			
	Value			Description
	0			APORT output enable controlled by IDAC_APORTOUTEN.
	1			APORT output enable controlled by PRS channel selected by PRSSEL.
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	APORTMASTERDIS	0	RW	**APORT Bus Master Disable**
	Determines if the IDAC will request the APORT bus selected by APORTOUTSEL. This bit allows multiple APORT connected devices to monitor the same APORT bus simultaneously by allowing the IDAC to not master the selected bus. When 1, the determination is expected to be from another peripheral, and the IDAC only passively looks at the bus. When 1, the selection of channel for a selected bus is ignored (the bus is not), and will be whatever selection the external device mastering the bus has configured for the APORT bus.			
	Value			Description
	0			Bus mastering enabled
	1			Bus mastering disabled
13	EM2DELAY	0	RW	**EM2 Delay**
	Delays EM2 entry until the IDAC output is stable			
12	PWRSEL	0	RW	**Power Select**
	Selects the power source for the IDAC			
	Mode	Value		Description
	ANA	0		AVDD
	IO	1		IOVDD
11:4	APORTOUTSEL	0x00	RW	**APORT Output Select**
	Select output mode.			
	APORT1XCH0	0x20		APORT1X Channel 0
	APORT1YCH1	0x21		APORT1Y Channel 1
	APORT1XCH2	0x22		APORT1X Channel 2
	APORT1YCH3	0x23		APORT1Y Channel 3
	APORT1XCH4	0x24		APORT1X Channel 4

Bit	Name	Reset	Access	Description
	APORT1YCH5	0x25		APORT1Y Channel 5
	. . .	. . .		. . .
	APORT1XCH30	0x3e		APORT1X Channel 30
	APORT1YCH31	0x3f		APORT1Y Channel 31
3	APORTOUTEN	0	RW	**APORT Output Enable**
	Set to enable the IDAC output to APORT if APORTOUTENPRS is not set.			
2	MINOUTTRANS	0	RW	**Minimum Output Transition Enable**
	Set to enable minimum output transition mode for the IDAC.			
1	CURSINK	0	RW	**Current Sink Enable**
	Set to enable the IDAC as a current sink. By default, the IDAC sources current.			
0	EN	0	RW	**Current DAC Enable**
	Set to enable the IDAC.			

29.5.2 IDAC_CURPROG - Current Programming 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												0x9B										0x00									0x0	
Access												RW										RW									RW	
Name												TUNING										STEPSEL									RANGESEL	

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	TUNING	0x9B	RW	**Tune the Current to Given Accuracy** In production test. the middle step (16) of each range is calibrated and can be read from the Device Information (DI) page.
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:8	STEPSEL	0x00	RW	**Current Step Size Select** Select the step within each range. The size of each step depends on the RANGESEL setting. RANGESEL settings of 0, 1, 2, and 3 correspond to step sizes of 50 nA, 100 nA, 500 nA, and 2000 nA, respectively. See step details.
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	RANGESEL	0x0	RW	**Current Range Select** Selects current range of the output.

Value	Mode	Description
0	RANGE0	Current range set to 0 - 1.6 uA.
1	RANGE1	Current range set to 1.6 - 4.7 uA.
2	RANGE2	Current range set to 0.5 - 16 uA.
3	RANGE3	Current range set to 2 - 64 uA.

29.5.3 IDAC_DUTYCONFIG - Duty Cycle Configuration 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																															0	
Access																															RW	
Name																															EM2DUTYCYCLEDIS	

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	EM2DUTYCYCLE-DIS	0	RW	**Duty Cycle Enable**
	Set to disable duty cycling in EM2.			
0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

29.5.4 IDAC_STATUS - 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																															0	0
Access																															R	R
Name																															APORTCONFLICT	CURSTABLE

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	APORTCONFLICT	0	R	**APORT Conflict Output**
	1 if any of the APORT BUSes being requested by the IDAC are also being requested by another peripheral			
0	CURSTABLE	0	R	**IDAC Output Current Stable**
	IDAC output current stable, after enabling or current programming.			

29.5.5 IDAC_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																															0	0
Access																															R	R
Name																															APORTCONFLICT	CURSTABLE

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	APORTCONFLICT	0	R	**APORT Conflict Interrupt Flag**
	1 if any of the APORT BUSes being requested by the IDAC are also being requested by another peripheral			
0	CURSTABLE	0	R	**Edge Triggered Interrupt Flag**
	Indicates that the IDAC output current is stable.			

29.5.6 IDAC_IFS - Interrupt Flag Set 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																															0	0
Access																															W1	W1
Name																															APORTCONFLICT	CURSTABLE

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	APORTCONFLICT	0	W1	**Set APORTCONFLICT Interrupt Flag**
	Write 1 to set the APORTCONFLICT interrupt flag			
0	CURSTABLE	0	W1	**Set CURSTABLE Interrupt Flag**
	Write 1 to set the CURSTABLE interrupt flag			

29.5.7 IDAC_IFC - Interrupt Flag Clear 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																															0	0
Access																															(R)W1	(R)W1
Name																															APORTCONFLICT	CURSTABLE

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	APORTCONFLICT	0	(R)W1	**Clear APORTCONFLICT Interrupt Flag**
	Write 1 to clear the APORTCONFLICT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	CURSTABLE	0	(R)W1	**Clear CURSTABLE Interrupt Flag**
	Write 1 to clear the CURSTABLE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

29.5.8 IDAC_IEN - Interrupt Enable 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																															0	0
Access																															RW	RW
Name																															APORTCONFLICT	CURSTABLE

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	APORTCONFLICT	0	RW	**APORTCONFLICT Interrupt Enable**
	Enable/disable the APORTCONFLICT interrupt			
0	CURSTABLE	0	RW	**CURSTABLE Interrupt Enable**
	Enable/disable the CURSTABLE interrupt			

29.5.9 IDAC_APORTREQ - APORT Request Status 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																													0	0		
Access																													R	R		
Name																													APORT1YREQ	APORT1XREQ		

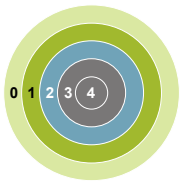
Bit	Name	Reset	Access	Description
31:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	APORT1YREQ	0	R	**1 If the Bus Connected to APORT1Y is Requested** Reports if the bus connected to APORT1Y is being requested by the APORT
2	APORT1XREQ	0	R	**1 If the APORT Bus Connected to APORT1X is Requested** Reports if the bus connected to APORT1X is being requested by the APORT
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

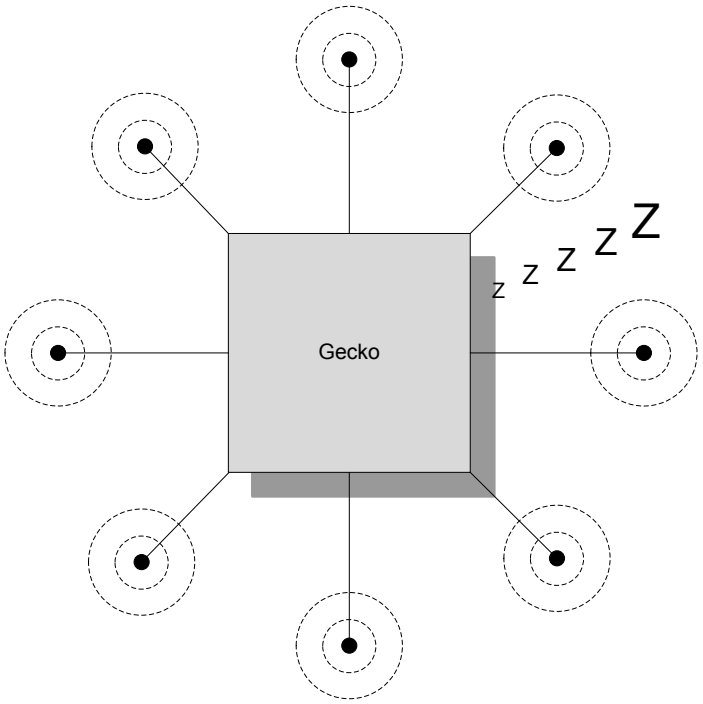
29.5.10 IDAC_APORTCONFLICT - APORT Request Status Register

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																													0	0		
Access																													R	R		
Name																													APORT1YCONFLICT	APORT1XCONFLICT		

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	APORT1YCONFLICT	0	R	**1 If the Bus Connected to APORT1Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT1Y is is also being requested by another peripheral
2	APORT1XCONFLICT	0	R	**1 If the Bus Connected to APORT1X is in Conflict With Another Peripheral** Reports if the bus connected to APORT1X is is also being requested by another peripheral
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

30. LESENSE - Low Energy Sensor Interface





Quick Facts

What?

LESENSE is a low energy sensor interface capable of autonomously collecting and processing data from multiple sensors even when in EM2. Flexible configuration makes LESENSE a versatile sensor interface compatible with a wide range of sensors and measurement schemes.

Why?

Capability to autonomously monitor sensors allows the EFM32 Giant Gecko 12 to reside in a low energy mode for long periods of time while keeping track of sensor status and sensor events.

How?

LESENSE is highly configurable and is capable of collecting data from a wide range of sensor types. Once the data is collected, the programmable state machine, LESENSE decoder, is capable of processing sensor data without CPU intervention. A large result buffer allows the chip to remain in EM2 for long periods of time while autonomously collecting data.

30.1 Introduction

LESENSE is a low energy sensor interface utilizing on-chip peripherals to perform measurement of a configurable set of sensors. The sensor measurements results can be processed by the LESENSE decoder, a configurable state machine with up to 32 states. The results can also be stored in a result buffer to be collected by the CPU or DMA for further processing.

LESENSE operates from EM0 down to EM2, and can wake up the CPU on configurable events.

30.2 Features

- Up to 16 sensors
- Autonomous sensor monitoring in EM0, EM1, and EM2
- Highly configurable decoding of sensor results
- Interrupt on sensor events
- Configurable enable signals to external sensors
- Circular buffer for storage of up to 16 sensor results
- Multiple evaluation modes minimize the need for software interaction
- Supports ADC0 sampling and evaluation
- Support for multiple sensor types
  - LC sensors
  - Capacitive sensing
  - General analog sensors

30.3 Functional Description

The LESENSE module is capable of controlling on-chip peripherals in order to perform monitoring of different sensors with little or no CPU intervention. LESENSE uses the analog comparators (ACMP) or ADC0 for measurement of sensor signals. LESENSE can also control the VDAC to generate accurate reference voltages. Figure 30.1 LESENSE Block Diagram on page 1098 shows an overview of the LESENSE module.

The LESENSE module consists of a sequencer, an evaluation block, a decoder, and a RAM block:

- The sequencer handles interaction with other peripherals and controls timing of sensor measurements. It also includes a counter that can be used to count pulses on the ACMP output.
- The evaluation block is used to process the data collected by the sequencer.
- To autonomously analyze sensor results, the LESENSE decoder provides the ability to define a finite state machine with up to 32 states, as well as define programmable actions upon state transitions. This allows the decoder to implement a wide range of decoding schemes, such as quadrature decoding.
- A RAM block is used for storage of configuration and measurement results. This allows LESENSE to have a relatively large result buffer enabling the chip to remain in a low energy mode for long periods of time while collecting sensor data.

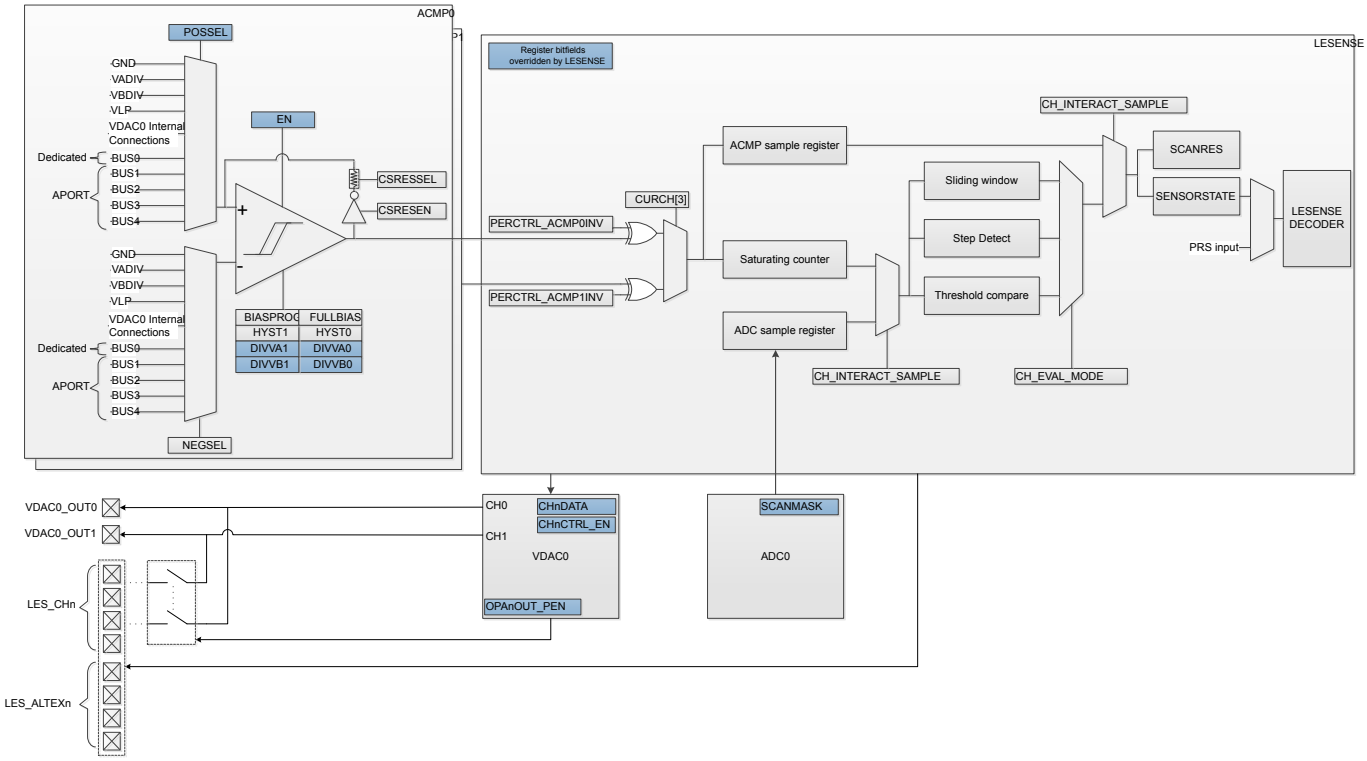


Figure 30.1. LESENSE Block Diagram

30.3.1 Channel Configuration

LESENSE has 16 individually configurable channels, each with its own set of configuration registers. Channel configuration is split into three registers; CHx_TIMING, CHx_INTERACT, and CHx_EVAL. Individual timing for each sensor is configured in CHx_TIMING, sensor interaction is configured in CHx_INTERACT, and configurations regarding evaluation of the measurements are done in CHx_EVAL. For improved readability, CHx_CONF will be used to refer to the channel configuration registers (CHx_TIMING, CHx_INTERACT, and CHx_EVAL) throughout this chapter.

By default, the channel configuration registers are directly mapped to the channel number. Configuring SCANCONF in CTRL makes it possible to alter this mapping.

Configuring SCANCONF to INVMAP will make channels 0-7 use the channel configuration registers for channels 8-15, and vice versa. This feature allows an application to quickly and easily switch the configuration set for the channels.

Setting SCANCONF to TOGGLE will make channel x alternate between using CH_X_CONF and CH_{X+8}_CONF. The configuration used is decided by the state of the corresponding bit in SCANRES. For instance, if channel 3 is performing a scan and bit 3 in SCANRES is set, CH_{11}_CONF will be used. Channels 8 through 15 will toggle between CH_X_CONF and CH_{X-8}_CONF. This mode provides an easy way to implement hysteresis on channel events, as threshold values can be changed depending on the sensor status.

Setting SCANCONF to DECDEF will make the state of the decoder define which scan configuration to be used. If the decoder state is at index 16 or higher, channel x will use CH_{X+8}_CONF, otherwise it will use CH_X_CONF. Similarly, channels 8 through 15 will use CH_X_CONF when the decoder state index is less than 8 and CH_{X-8}_CONF when the decoder state index is higher than 7. Allowing the decoder state to define which configuration to use enables easy implementation of hysteresis, for example, as different threshold values can be used for the same channel depending on the state of the application. Table 30.1 LESENSE Scan Configuration Selection on page 1099 summarizes how channel configuration is selected for different settings of SCANCONF.

Table 30.1. LESENSE Scan Configuration Selection

LESENSE channel x	SCANCONF					
	DIRMAP	INVMAP	TOGGLE		DECDEF	
			SCANRES[n] = 0	SCANRES[n] = 1	DECSTATE < 16	DECSTATE >= 16
0 <= x < 8	CH_x_CONF	CH_{x+8}_CONF	CH_x_CONF	CH_{x+8}_CONF	CH_x_CONF	CH_{x+8}_CONF
8 <= x < 16	CH_x_CONF	CH_{x-8}_CONF	CH_x_CONF	CH_{x-8}_CONF	CH_x_CONF	CH_{x-8}_CONF

Channels are enabled in the CHEN register, where bit x enables channel x. During a scan, all enabled channels are measured, starting with the lowest indexed channel. Figure 30.3 Scan Sequence on page 1100 illustrates a scan sequence with channels 3, 5, and 9 enabled.

30.3.2 Scan Sequence

LESENSE runs on $LFACLK_{LESENSE}$, which is a prescaled version of LFACLK. The prescaling factor for $LFACLK_{LESENSE}$ is selected in the CMU, available prescaling factors are:

- DIV1: $LFACLK_{LESENSE}$ = LFACLK/1
- DIV2: $LFACLK_{LESENSE}$ = LFACLK/2
- DIV4: $LFACLK_{LESENSE}$ = LFACLK/4
- DIV8: $LFACLK_{LESENSE}$ = LFACLK/8

All enabled channels are scanned each scan period. How a new scan is started is configured in the SCANMODE bit field in CTRL. If set to PERIODIC, the scan frequency is generated using a counter which is clocked by $LFACLK_{LESENSE}$. This counter has its own prescaler. This prescaling factor is configured in PCPRESC in TIMCTRL. A new scan sequence is started each time the counter reaches the top value, PCTOP. The scan frequency is calculated using Figure 30.2 Scan Frequency on page 1100. If SCANMODE is set to ONESHOT, a single scan will be made when START in CMD is set. To start a new scan on a PRS event, set SCANMODE to PRS and configure PRS channel in PRSSEL. The PRS start signal needs to be active for at least one $LFACLK_{LESENSE}$ cycle to make sure LESENSE is able to register it.

$$F_{scan} = LFACLK_{LESENSE}/((1 + PCTOP)*2^{PCPRESC})$$

Figure 30.2. Scan Frequency

It is possible to interleave additional sensor measurements in between the periodic scans. Issuing a start command when LESENSE is idle will immediately start a new scan, without disrupting the frequency of the periodic scans. If the period counter overflows during the interleaved scan, the periodically scheduled scan will start immediately after the interleaved scan completes.

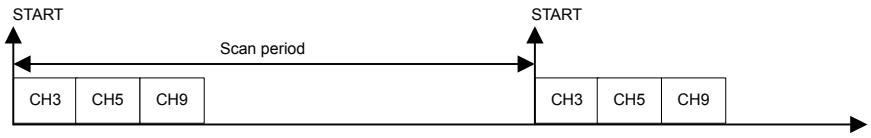


Figure 30.3. Scan Sequence

30.3.3 Sensor Timing

For each channel in the scan sequence, the LESENSE interface goes through three phases: idle, excite, and measure. The durations of the excite and measure phases are configured in the CHx_TIMING registers. The excite phase duration can be configured to be either a number of AUXHFRCO cycles or a number of $LFACLK_{LESENSE}$ cycles, depending on which one is selected by the EXCLK bit in the CHx_INTERACT register. LESENSE includes two timers: A low frequency timer, running on $LFACLK_{LESENSE}$, and a high frequency timer, running on AUXHFRCO. The low frequency or high frequency timers can be prescaled by configuring LFPRESC or AUXPRESC, respectively, in the TIMCTRL register. The duration of the measure phase is programmed via MEASUREDLY and SAMPLEDLY in the CHx_TIMING registers. The output of the ACMP will be ignored for MEASUREDLY EXCLK cycles after start of the sensor measurement. Sampling of the sensor will happen after SAMPLEDLY $LFACLK_{LESENSE}$, or AUXHFRCO cycles, depending on the configuration of the SAMPLECLK in the CHx_INTERACT register. The configurable measure- and sample delays enables LESENSE to easily define exact time windows for sensor measurements. A start delay can be inserted before sensor measurement begin by configuring STARTDLY in TIMCTRL. This delay can be used to ensure that the VDAC conversion is done and voltages have stabilized before the sensor measurement begins. The AUXHFRCO startup can be delayed until the system enters the excite phase, by configuring AUXSTARTUP in TIMCTRL to ONDEMAND. This will reduce the time the AUXHFRCO is enabled and reduce power consumption, with the tradeoff that that the starting point for high frequency timing will also be delayed the same amount as the AUXHFRCO startup time.

Figure 30.4 Timing Diagram, AUXHFRCO Based Timing on page 1101 depicts a sensor sequence with AUXHFRCO based timing (EXTIME=5, MEASUREDLY=7, SAMPLEDLY=13), while Figure 30.5 Timing Diagram, LFACLK Based Timing on page 1102 depicts a sequence with $LFACLK_{LESENSE}$ based timing (EXTIME=1, MEASUREDLY=1, SAMPLEDLY=2).

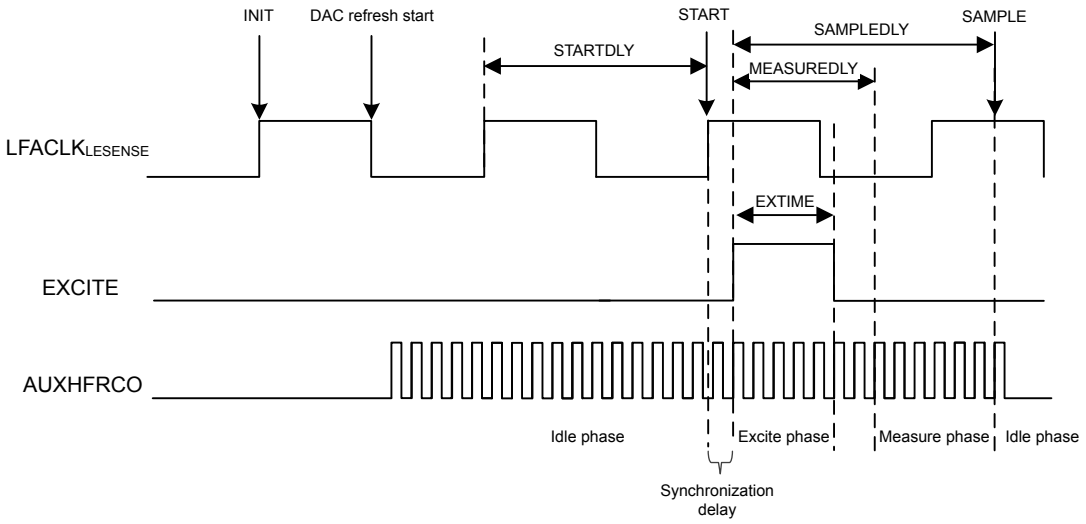


Figure 30.4. Timing Diagram, AUXHFRCO Based Timing

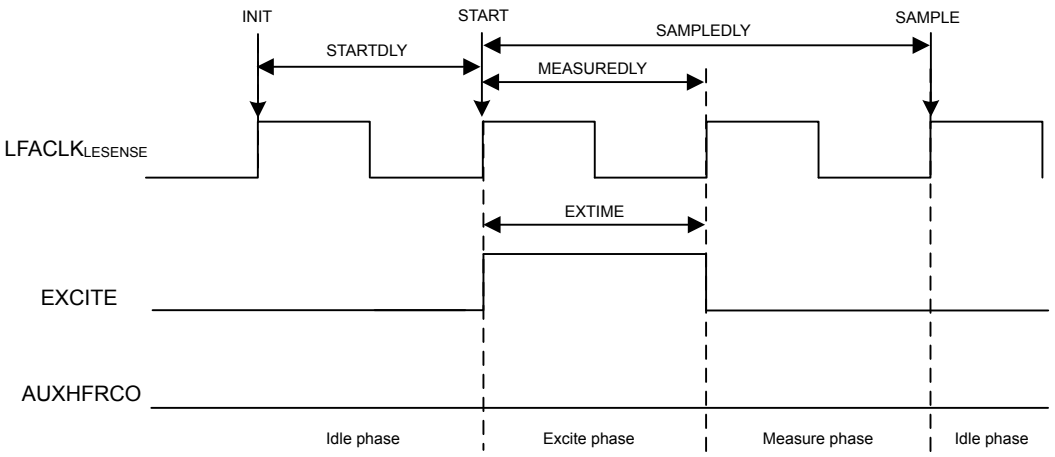


Figure 30.5. Timing Diagram, LFACLK Based Timing

30.3.4 Sensor Interaction

Many sensor types require some type of excitation in order to work. The LESENSE module can generate a variety of sensor stimuli, both on the same pin as the measurement is to be made on, as well as alternative pins.

By default, excitation is performed on the pin associated with the channel (i.e., excitation and sensor measurement is performed on the same pin). The mode of the pin during the excitation phase is configured by the EXMODE bitfield in CHx_INTERACT. The available modes during the excite phase are:

- DISABLED: The pin is disabled.
- HIGH: The pin is driven high.
- LOW: The pin is driven low.
- DACOUT: The pin is connected to the output of a VDAC channel.

Note: Excitation with VDAC output is only available on some channels. Refer to 30.3.9 VDAC Interface for details. If the VDAC is in opamp-mode, setting EXMODE to DACOUT will result in excitation with output from the opamp.

LESENSE is able to perform sensor excitation on a pin other than the one being measured. When ALTEX in CHx_INTERACT is set, the excitation will occur on the alternative excite pin associated with the given channel. By default, the alternative excite pins are mapped to the LES_ALTEX pins, but they can also be mapped to LESENSE $CH_{X+8 \bmod 16}$. Mapping of the alternative excite pins is configured in ALTEXMAP in the CTRL register. Table 30.2 LESENSE Excitation Pin Mapping on page 1103 summarizes the mapping of excitation pins for different configurations.

Table 30.2. LESENSE Excitation Pin Mapping

LESENSE channel	ALTEX = 0	ALTEX = 1	
		ALTEXMAP = CH	ALTEXMAP = ALTEX
0	LES_CH0	LES_CH8	LES_ALTEX0
1	LES_CH1	LES_CH9	LES_ALTEX1
2	LES_CH2	LES_CH10	LES_ALTEX2
3	LES_CH3	LES_CH11	LES_ALTEX3
4	LES_CH4	LES_CH12	LES_ALTEX4
5	LES_CH5	LES_CH13	LES_ALTEX5
6	LES_CH6	LES_CH14	LES_ALTEX6
7	LES_CH7	LES_CH15	LES_ALTEX7
8	LES_CH8	LES_CH0	LES_ALTEX0
9	LES_CH9	LES_CH1	LES_ALTEX1
10	LES_CH10	LES_CH2	LES_ALTEX2
11	LES_CH11	LES_CH3	LES_ALTEX3
12	LES_CH12	LES_CH4	LES_ALTEX4
13	LES_CH13	LES_CH5	LES_ALTEX5
14	LES_CH14	LES_CH6	LES_ALTEX6
15	LES_CH15	LES_CH7	LES_ALTEX7

Figure 30.6 Pin Sequencing on page 1104 illustrates the sequencing of the pin associated with the active channel and its alternative excite pin.

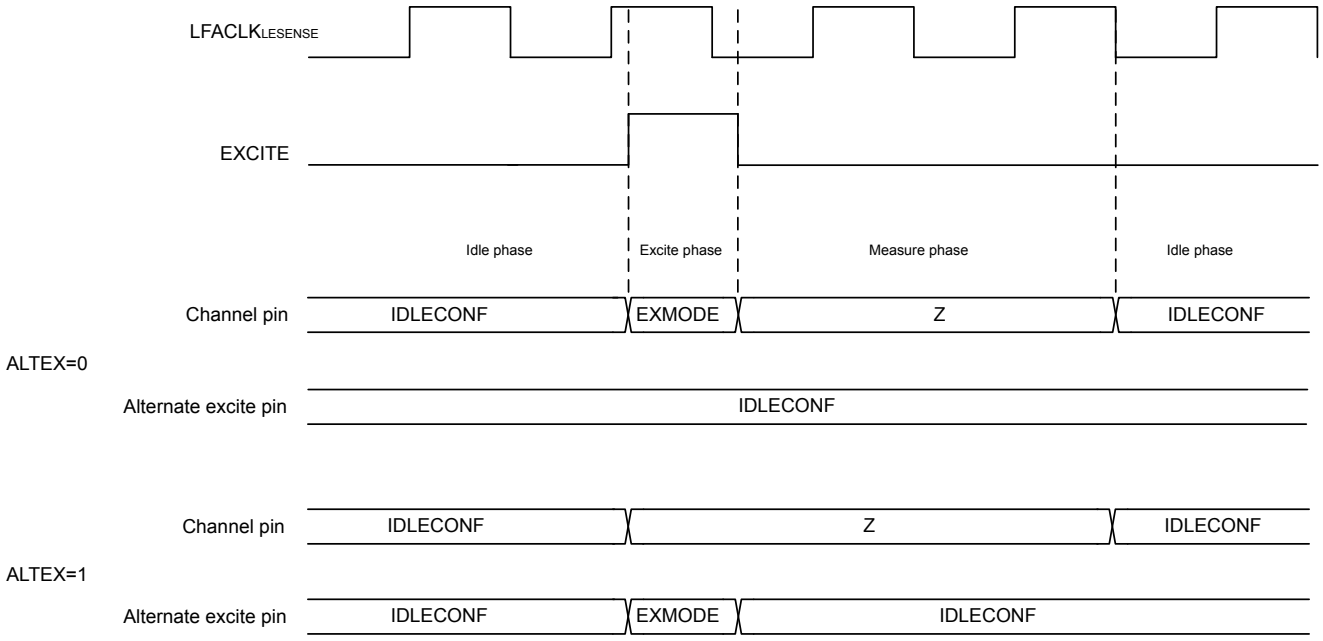


Figure 30.6. Pin Sequencing

The LES_ALTEXn pins have the ability to excite regardless of what channel is active. Setting AEXn in ALTEXCONF will make LES_ALTEXn excite for all channels using alternative excitation (i.e., ALTEX in CHx_INTERACT is set).

Note: When exciting on the pin associated with the active channel, the pin will go through a tri-stated phase before returning to the idle configuration. This will not happen on pins used as alternative excitation pins.

The pin configuration for the idle phase can be configured individually for each LESENSE channel and alternative excite pin in the IDLECONF and ALTEXCONF registers. The modes available are the same as the modes available in the excitation phase. In the measure phase, the pin mode on the active channel is always disabled (analog input).

To allow the LESENSE mode to control a GPIO pin, the pin must be enabled in the ROUTEPEN register and configured as push-pull. The IDLECONF configuration should not be altered while the pin enable for a given pin is set in ROUTEPEN.

30.3.5 Sensor Sampling

During the measurement phase, LESENSE can sample data from sensors using either ADC0 or an ACMP. This is configured in CHx_INTERACT_SAMPLE. If the ACMP is used, LESENSE can evaluate the ACMP output at a single point in time (CHx_INTERACT_SAMPLE = ACMP), or count pulses on the ACMP output (CHx_INTERACT_SAMPLE = ACMPCOUNT) for a programmable period of time.

LESENSE includes the ability to sample both analog comparators simultaneously, effectively cutting the time spent on sensor interaction in some applications in half. Setting DUALSAMPLE in CTRL enables this mode. In dual sample mode, channels X and X+8 are paired, meaning they will be sampled at the same time. DUALSAMPLE mode only works when CHx_INTERACT_SAMPLE is set to ACMP.

If ADC0 is used, LESENSE will initiate ADC conversions and fetch the ADC data for further evaluation. If the ADC is configured in differential mode, CHx_INTERACT_SAMPLE must be set to ADCDIFF. In this mode, the output from the ADC and the threshold used for comparison are given in two's complement notation.

See sections30.3.12 ADC Interface and 30.3.10 ACMP Interface for more details on the LESENSE interface to the ADC and ACMPs. The sampled data from ADC or ACMP will be referred to as sensor data in the remainder of this manual.

30.3.6 Sensor Evaluation

When a measurement phase is completed, the sensor data is evaluated by the evaluation block. If the sensor data is taken from ACMP sample in a single point in time (CHx_INTERACT_SAMPLE = ACMP), the evaluation is limited to determining if the sensor data is 0 or 1. For the other sample modes, there are three ways to do sensor evaluation; threshold comparison, sliding window, or step detection. Evaluation mode is configured in CHx_EVAL_MODE.

If the evaluation of sensor data evaluates to true, the corresponding bit in the result register (SCANRES) is set. By configuring SETIF in CHx_INTERACT, interrupt flags can also be set on SCANRES events. Figure 30.7 Scan Result and Interrupt Generation on page 1105 illustrates how the sensor data or ACMP sample is used for evaluation and interrupt generation.

Note: For initialization purposes, SCANRES can be written by software. SCANRES should not be written while LESENSE is running (i.e., the RUNNING bit in LESENSE_STATUS is high).

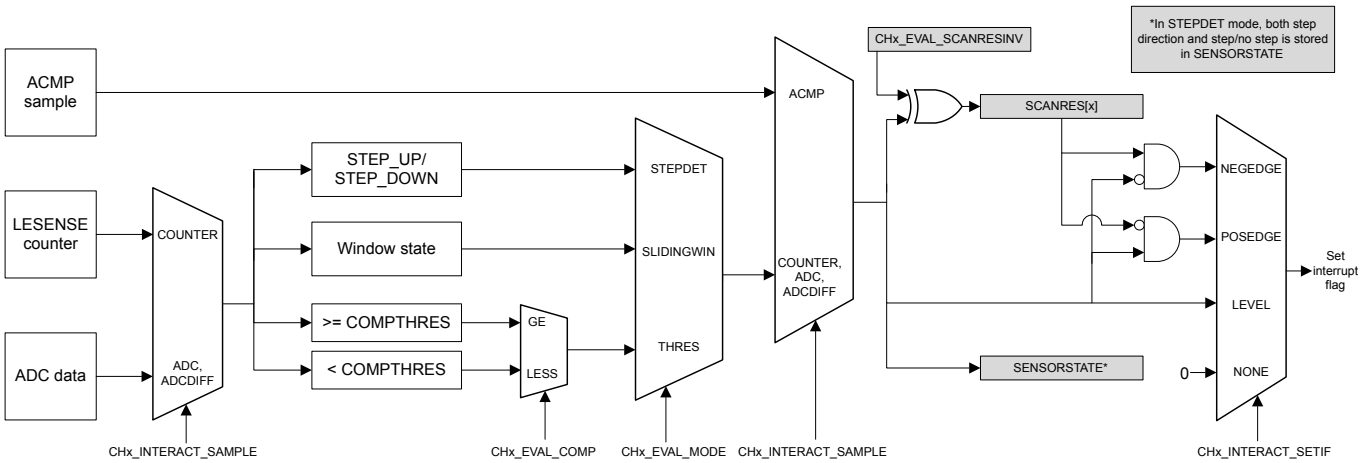


Figure 30.7. Scan Result and Interrupt Generation

The results from sensor data evaluation can be fed into the decoder through the SENSORSTATE register. In DUALSAMPLE mode, results from both the sampled ACMPs will be stored in both SCANRES and SENSORSTATE.

30.3.6.1 Threshold Comparison

In threshold comparison mode, the sensor data is compared to a threshold configured in CHx_EVAL_COMPTHRES. There are two modes of threshold comparison: 'less than' and 'greater than or equal'. Threshold comparison mode is configured in CHx_EVAL_COMP.

30.3.6.2 Sliding Window

In sliding window mode, the sensor data is compared against the upper and lower limits of a window range. The window is defined by a base, given by CHx_EVAL_COMPTHRES, and a size configured in EVALCTRL_WINSIZE. The window size is constant and the same for all LESENSE channels, while the base is specific to each channel and will be updated by LESENSE when the sensor data is outside the current window range. If the sensor data is within the window range, the sensor evaluation will remain the same as it was for the previous measurement. If the sensor data is below the window range, the measurement will be evaluated to false. If the sensor data is above the window range, the measurement will be evaluated to true. In both cases, the window base in CHx_EVAL_COMPTHRES will be updated to reflect the new window range. Figure 30.8 Sliding Window on page 1106 shows how the sliding window evaluation mode can be used to implement a system with two self calibrating thresholds.

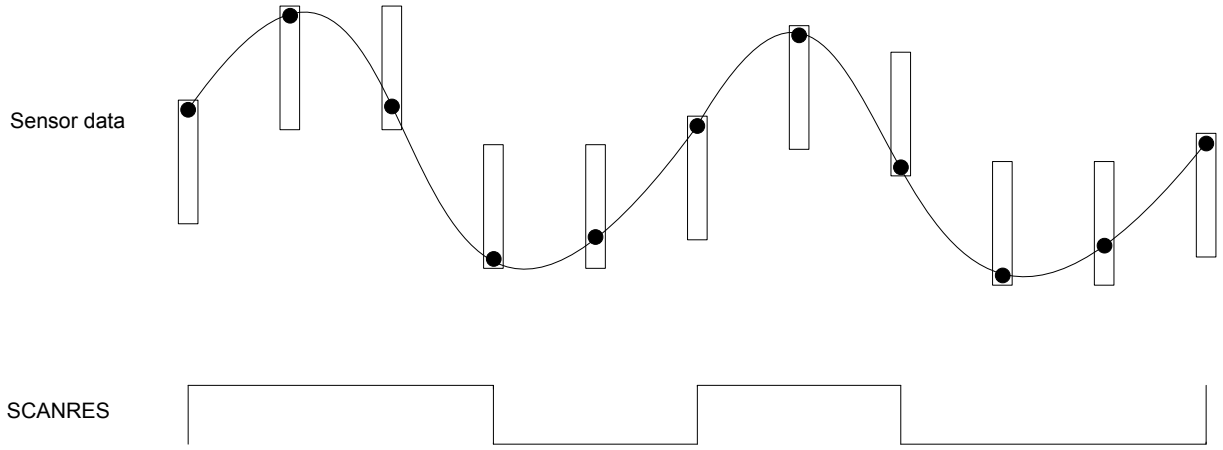


Figure 30.8. Sliding Window

30.3.6.3 Step Detection

Step detection is used to detect steps in the sensor data compared to sensor data from the previous measurement. The size of the step is configured in EVALCTRL_WINSIZE. In this mode, step up and step down are evaluated as described in Figure 30.9 Step Detection on page 1106:

$$STEP\_UP = SENSORDATA_i >= SENSORDATA_{i-1} + EVALCTRL\_WINSIZE$$

$$STEP\_DOWN = SENSORDATA_i < SENSORDATA_{i-1} - EVALCTRL\_WINSIZE$$

Figure 30.9. Step Detection

If either a step up or a step down is detected, the SCANRES bit for the active channel will be set. In addition, the STEPDIR bit for the channel will be updated to indicate if a step up or a step down was detected. STEPDIR = 1 indicates a step up. In this mode, previous sensor data is stored in CHx_EVAL_COMPTHRES.

30.3.7 Decoder

Many applications, such as quadrature decoding, require some sort of processing of the sensor readings. In quadrature decoding, the sensors repeatedly pass through a set of states which correspond to the position of the sensors. This sequence, and many other decoding schemes, can be described as a finite state machine. To support this type of decoding without CPU intervention, the LESENSE module includes a highly configurable decoder capable of decoding input from up to four sensors. The decoder is implemented as a programmable state machine with up to 32 states. When doing a sensor scan, the results from the sensors are placed in the decoder input register, SENSORSTATE, if DECODE in CHx_INTERACT is set. The resulting position after a scan is illustrated in Figure 30.10 Sensor Scan and Decode Sequence on page 1107, where the bottom blocks show how the SENSORSTATE register is filled. If step detection is enabled, the step direction is placed in SENSORSTATE in the position after the sensor result. When the scan sequence is complete, the decoder evaluates the state of the sensors chosen for decoding, as depicted in Figure 30.10 Sensor Scan and Decode Sequence on page 1107.

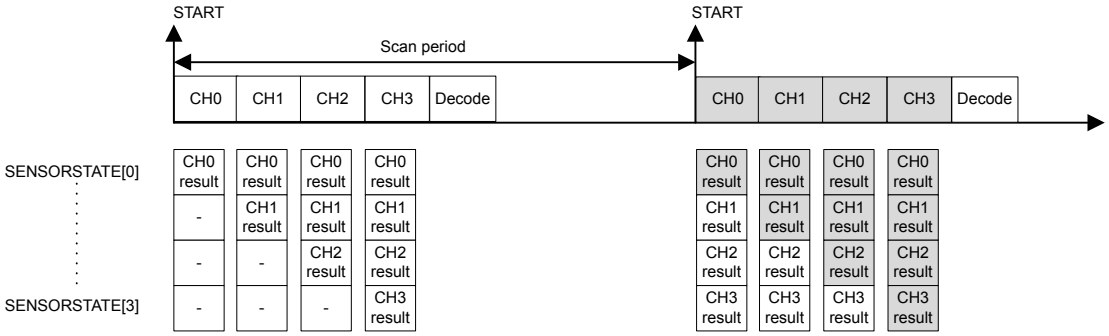


Figure 30.10. Sensor Scan and Decode Sequence

Upon a state transition, LESENSE can generate a pulse on one or more of the decoder PRS channels. Which PRS channel to generate a pulse on is configured in the PRSACT bit field. If PRSCNT in DECCTRL is set, count signals will be generated on decoder PRS channels 0 and 1 according to the PRSACT configuration. In this mode, channel 0 will pulse each time a count event occurs, while channel 1 indicates the count direction (1 being up and 0 being down). The count direction will be kept at its previous state in between count events. The EFM32 Giant Gecko 12 pulse counter may be used to keep track of events based on these PRS outputs.

If SETIF is set, the DECODER interrupt flag will be set when the transition occurs. If INTMAP in DECCTRL and SETIF is set, a transition from state x or x+16 will set the CHx interrupt flag in addition to the DECODER flag.

Setting CHAIN in STx_TCONFA enables the decoder to evaluate more than two possible transitions for each state. If none of the transitions defined in STx_TCONFA or STx_TCONFB match, the decoder will jump to the next descriptor pair and evaluate the transitions defined there. The decoder uses two $LFACLK_{LESENSE}$ cycles for each descriptor pair to be evaluated. If ERRCHK in CTRL is set, the decoder will check that the sensor state has not changed if none of the defined transitions match. The DECERR interrupt flag will be set if none of the transitions match and the sensor state has changed. Figure 30.11 Decoder State Transition Evaluation on page 1108 illustrates state transitions. The "Generate PRS signals and set interrupt flag" blocks will perform actions according to the configuration in STx_TCONFA and STx_TCONFB.

Note: If only one transition from a state is used, STx_TCONFA and STx_TCONFB should be configured equally.

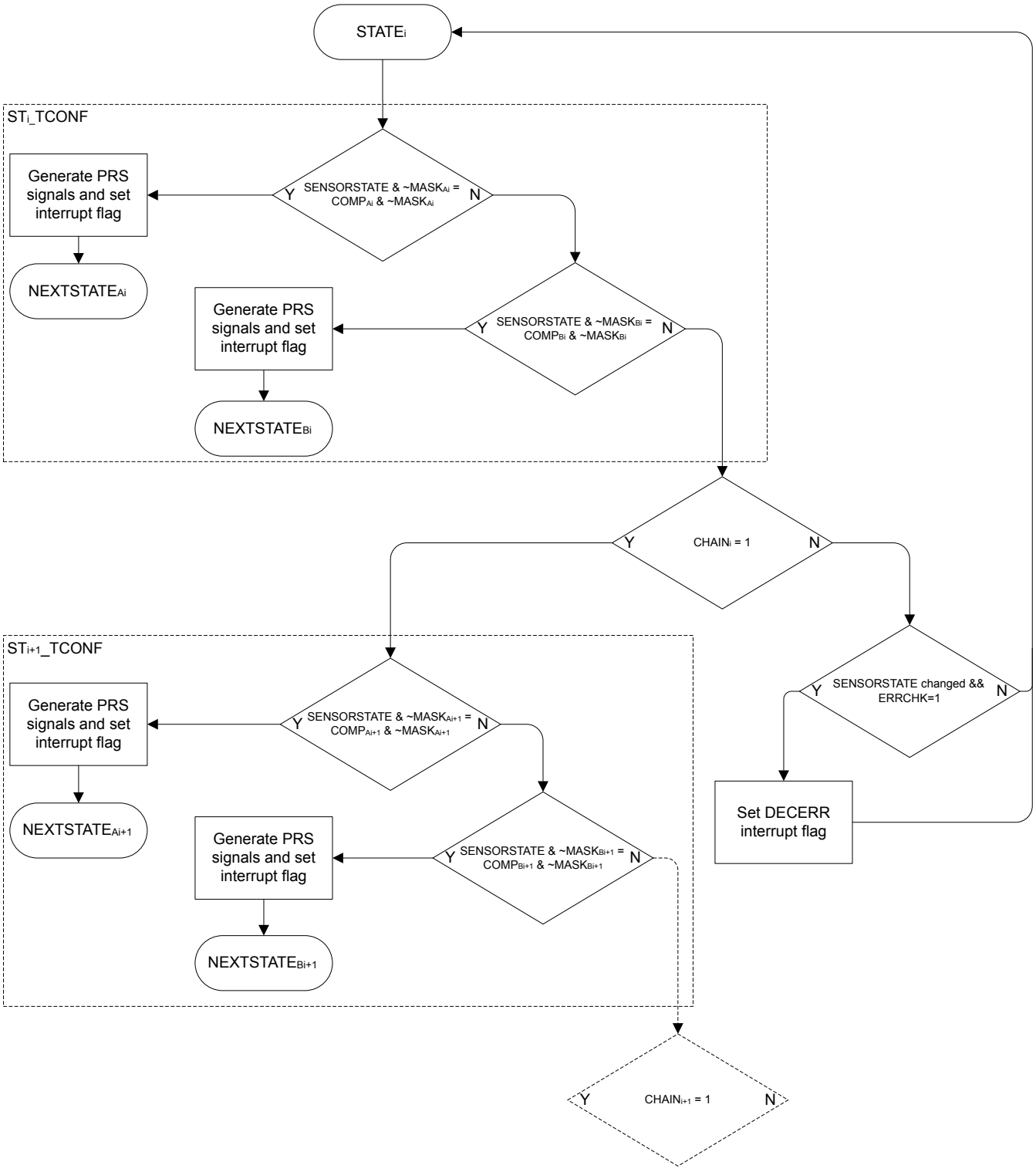


Figure 30.11. Decoder State Transition Evaluation

The DECODER has a PRS output named DECCMP. This output can be used to indicate which state, or subset of states, the decoder is currently in. This PRS output is enabled by setting DECCMPEN in PRSCTRL, and configured through DECCMPMASK and DECCMPVAL in PRSCTRL. The value of this PRS output is given by Figure 30.12 DECCMP PRS Output on page 1109,

PRS_DECCMP = (DECSTATE & ~DECCMPMASK) == (DECCMPVAL & ~DECCMPMASK)

Figure 30.12. DECCMP PRS Output

To prevent unnecessary interrupt requests or PRS outputs when the decoder toggles back and forth between two states, a hysteresis option is available. The hysteresis function is triggered if a type A transition is preceded by a type B transition, and vice versa. A type A transition is defined in STx_TCONFA, and a type B transition is defined in STx_TCONFB. When descriptor chaining is used, a jump to another descriptor will cancel out the hysteresis effect. Figure 30.13 Decoder Hysteresis on page 1109 illustrates how the hysteresis triggers upon state transitions.

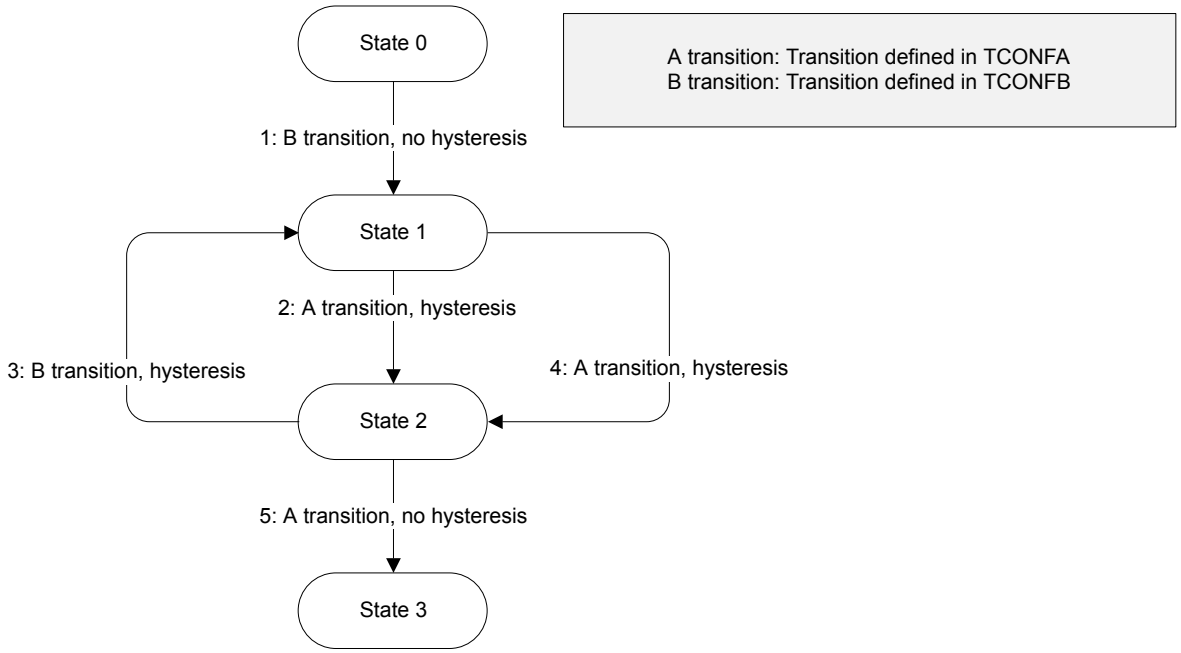


Figure 30.13. Decoder Hysteresis

- When HYSTPRSx is set, PRS signal x is suppressed when the hysteresis triggers.
- When HYSTIRQ is set, interrupt requests are suppressed when the hysteresis triggers.

Note: The decoder error interrupt flag, DECERR, is not affected by the hysteresis.

30.3.8 Measurement Results

Part of the LESENSE RAM is treated as a circular buffer for storage of up to 16 sensor measurements results. Each time LESENSE writes data to the result buffer, the result write pointer (PTR_WR) is incremented. Each time a new result is read through the BUFDATA register, the result read pointer (PTR_RD) is incremented. The read pointer will not be incremented if there is no valid, unread data in the result buffer. By default LESENSE will not write additional data to a full result buffer until the data is read by software or DMA. Setting BUFOW in CTRL enables LESENSE to write to the result buffer even if it is full. In this mode, the result read pointer will follow the write pointer if the buffer is full. The result of this is that data read from the result read register (BUFDATA) will be the oldest unread result. The location pointers are available in PTR.

The result buffer has three flags in the STATUS register: BUFDATAV, BUFHALFFULL, and BUFFULL. The flags indicate when new data is available, when the buffer is half full, and when it is full, respectively.

The result buffer also has three interrupt flags in the IR register: BUFDATAV, BUFLEVEL, and BUFOF. BUFDATAV is set when data is available in the buffer. BUFLEVEL is set when the buffer is either full or half-full, depending on the configuration of BUFIDL in CTRL. BUFOF is set if the result buffer overflows.

During a scan, the state of each sensor is stored in SCANRES. If a sensor triggers, a 1 is stored in SCANRES, else a 0 is stored in SCANRES. Whether or not a sensor is said to be triggered depends of the configuration for the given channel. See 30.3.6 Sensor Evaluation for details. If STRSAMPLE in CHx_EVAL is set, the sensor data for each channel will be stored in the LESENSE result buffer. If STRSCANRES in CTRL is set, the result vector, SCANRES, will also be stored in the result buffer. This will be stored after each scan and will be interleaved with the counter values. The contents of the result buffer can be read from BUFDATA or from BUF[x]_DATA. When reading from BUF[x]_DATA, neither the result read pointer or the status flags BUFDATAV, BUFHALFFULL, or BUFFULL will be updated. When reading through the BUFDATA register, the oldest unread result will be read.

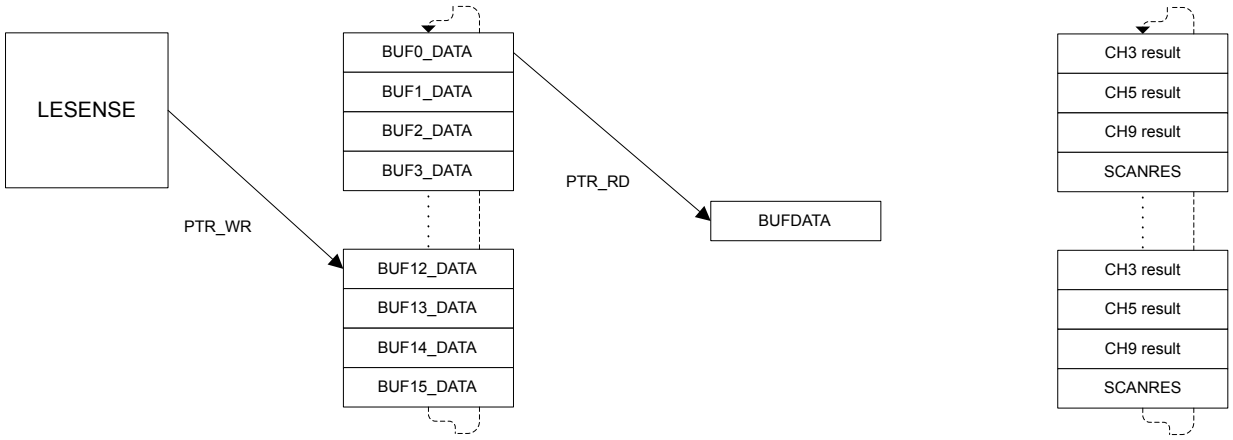


Figure 30.14. Circular Result Buffer

Figure 30.14 Circular Result Buffer on page 1110 illustrates how the result buffer would be filled when channels 3,5, and 9 are enabled and have STRSAMPLE in CHx_EVAL set, in addition to STRSCANRES in CTRL. The measurement result from the three channels will be sequentially written during the scan, while SCANRES is written to the result buffer upon scan completion.

30.3.9 VDAC Interface

LESENSE is able to drive the VDAC for generation of accurate reference voltages. This is enabled by setting DACCHxEN in PERCTRL. The refresh rate of the VDAC channels can be configured in DACCONVTRIG in PERCTRL. If DACCONVTRIG is set to CHANNELSTART, the VDAC channels are refreshed prior to each sensor measurement, as depicted in Figure 30.4 Timing Diagram, AUXHFRCO Based Timing on page 1101. If DACCONVTRIG is set to SCANSTART, the VDAC channels are refreshed prior to each scan. The conversion data is either taken from the data registers in the EFM32 Giant Gecko 12 VDAC interface (VDAC0_CH0DATA and VDAC0_CH1DATA) or from the THRES bitfield in the CHx_INTERACT register for the active LESENSE channel. VDAC data used is configured in DACCHxDATA in PERCTRL.

Bias configuration, calibration and reference selection is done in the EFM32 Giant Gecko 12 VDAC module and LESENSE will not override these configurations.

LESENSE has the possibility to control switches that connect the VDAC alternate outputs. This allows LESENSE to excite sensors with output from the VDAC channels, this is done by setting CHx_INTERACT_EXMODE to DACOUT. The LESENSE channels can also be connected to the VDAC output when the given channel is idle, this is done by setting IDLECONF_CHx to DAC.

Note: Only LESENSE channels 0, 1, 2, 3, 12, 13, 14, 15 have the possibility to excite using the VDAC alternate outputs, or connect to the VDAC alternate outputs during the idle phase.

The VDAC may be chosen as reference to the analog comparators for accurate reference generation. If the VDAC is configured in continuous mode this does not require any external components. If sample/off mode is used, an external capacitor is needed to maintain the voltage between samples. To configure the VDAC to use this external capacitor, connect the capacitor to the VDAC pin for the given channel and set SHORT in VDAC_OPAx_OUT.

Note: The VDAC mode should not be altered while DACACTIVE in STATUS is set

30.3.10 ACMP Interface

The analog comparators (ACMPs) are used to measure the sensors, and have to be configured according to the application in order for LESENSE to work properly. Depending on the configuration in the ACMP0MODE and ACMP1MODE bit-fields in PERCTRL, LESENSE will take control of the positive input mux and the voltage dividers (DIVVA, DIVVB) for ACMP0 and ACMP1. The remaining configuration of the analog comparators is done in the ACMP register interface.

If ACMPxMODE in PERCTRL is set to MUX, LESENSE will take control of the positive input mux of the ACMP, through the external override interface, described in the ACMP chapter (see 27.3.12 External Override Interface). The offset given by LESENSE, EXT_OFFSET, depends on whether one or two ACMPs are controlled by LESENSE. If only one ACMP is used, EXT_OFFSET will have the same value as the active channel. If both ACMP0 and ACMP1 are used, LESENSE channel 0-7 will use ACMP0 with EXT_OFFSET 0-7, and LESENSE channel 8-15 will use ACMP1 with EXT_OFFSET 0-7.

If ACMPxMODE in PERCTRL is set to MUXTHRES, LESENSE will also take control of the voltage dividers in the ACMP, DIVVA and DIVVB. The thresholds used are individual to each channel and is configured using the 6 LSBs of CHx_INTERACT_THRES. By default, ACMP_HYSTERESIS0_DIVVX and ACMP_HYSTERESIS1_DIVVX will be given the same value, the 6 LSBs of CHx_INTERACT_THRES. To allow different values for ACMP_HYSTERESIS0_DIVVX and ACMP_HYSTERESIS1_DIVVX, ACMPxHYSTEN in PERCTRL needs to be set. This allows the hysteresis feature in the ACMP to be utilized. ACMP_HYSTERESIS0_DIVVX will get the value programmed in CHx_INTERACT_THRES[5:0], while ACMP_HYSTERESIS1_DIVVX will get the value programmed in CHx_INTERACT_THRES[11:6].

30.3.11 ACMP and VDAC Duty Cycling

By default, the analog comparators and the VDAC are shut down between LESENSE scans to save energy. If this is not desired, WARMUPMODE in PERCTRL can be configured to prevent them from being shut down.

Both the VDAC and analog comparators rely on a bias module for correct operation. This bias module has a low power mode which consumes less energy at the cost of reduced accuracy. BIASMODE in BIASCTRL configures how the bias module is controlled by LESENSE. When set to DUTYCYCLE, LESENSE will set the bias module in high accuracy mode whenever LESENSE is active, and keep it in the low power mode otherwise. When BIASMODE is set to HIGHACC, the high accuracy mode is always selected. When set to DONTTOUCH, LESENSE will not control the bias module.

30.3.12 ADC Interface

The LESENSE module can be configured to trigger conversions on ADC0 and use data from ADC0 to evaluate sensor status. In order to do this, the scan mode of the ADC has to be configured. When the sample delay configured in CHx_TIMING_SAMPLEDLY has expired, LESENSE will initiate an ADC sample. The active LESENSE channel determines which ADC0 channel to be sampled, where LESENSE channel X corresponds to ADC0 scan channel X.

30.3.13 DMA Requests

LESENSE issues a DMA request when the result buffer is either full or half full, depending on the configuration of BUFIDL in CTRL. The request is cleared when the buffer level drops below the threshold defined in BUFIDL. A single DMA request is also set whenever there is unread data in the buffer. DMAWU in CTRL configures at which buffer level LESENSE should wake-up the DMA when in EM2.

Note: The DMA controller should always fetch data from the BUFDATA register.

30.3.14 PRS Output

LESENSE is an asynchronous PRS producer and has twenty PRS outputs. The decoder has four outputs and in addition, all bits in the SCANRES register are available as PRS outputs. For further information on the decoder PRS output, refer to 30.3.7 Decoder.

30.3.15 RAM

LESENSE includes a RAM block used for storage of configuration and results. Registers mapped to the RAM include: STx_TCONFA, STx_TCONFB, BUFx_DATA, BUFDATA, CHx_TIMING, CHx_INTERACT, and CHx_EVAL. These registers have unknown value out of reset and have to be initialized before use.

Note: Read-modify-write operations on uninitialized RAM register produces undefined values.

30.3.16 Application Examples

The following sections detail several example applications for the LESENSE block.

30.3.16.1 Capacitive Sense

Figure 30.15 Capacitive Sense Setup on page 1113 illustrates how the EFM32 Giant Gecko 12 can be configured to monitor four capacitive buttons.

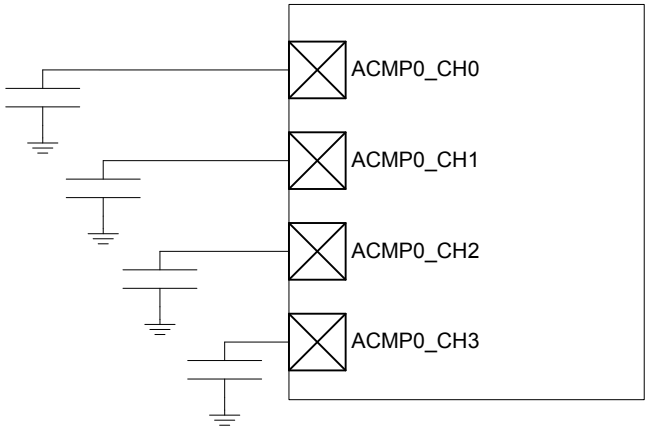


Figure 30.15. Capacitive Sense Setup

The following steps show how to configure LESENSE to scan through the four buttons 100 times per second, issuing an interrupt if one of them is pressed.

1. Assuming $LFACLK_{LESENSE}$ is 32 kHz, set PCPRESC to 3 and PCTOP to 39 in CTRL. This will set the LESENSE scan frequency to 100 Hz.
2. Enable channels 0 through 3 in CHEN and set IDLECONF for these channels to DISABLED. In capacitive sense mode, the GPIO should always be disabled (i.e., analog input).
3. Configure the ACMP to operate in CAPSENSE mode (refer to the ACMP chapter for more details).
4. Configure the following bit fields in CHx_CONF, for channels 0 through 3:
   a. Set EXTIME to 0. No excitation is needed in this mode.
   b. Set SAMPLE to ACMPCOUNT and COMP to LESS. This makes LESENSE interpret a sensor as active if the frequency on a channel drops below the threshold (i.e., the button is pressed).
   c. Set SAMPLEDLY to an appropriate value - each sensor will be measured for SAMPLEDLY/$F_{LFACLK\_LESENSE}$ seconds. MEASUREDLY should be set to 0
5. Set CTRTHRESHOLD to an appropriate value. An interrupt will be issued if the counter value for a sensor is below this threshold after the measurement phase.
6. Enable interrupts on channels 0 through 3.
7. Start scan sequence by writing a 1 to START in CMD.

In a capacitive sense application, it might be required to calibrate the threshold values on a periodic basis, for example to compensate for humidity and other physical variations. LESENSE is able to store up to 16 counter values from a configurable number of channels, making it possible to collect sample data while in EM2. When calibration is to be performed, the CPU only has to be woken up for a short period of time as the data to be processed already lies in the result registers. To enable storing of the count value for a channel, set STRSAMPLE in the CHx_INTERACT register.

30.3.16.2 LC Sensor

Figure 30.16 LC Sensor Setup on page 1114 below illustrates how the EFM32 Giant Gecko 12 can be set up to monitor four LC sensors.

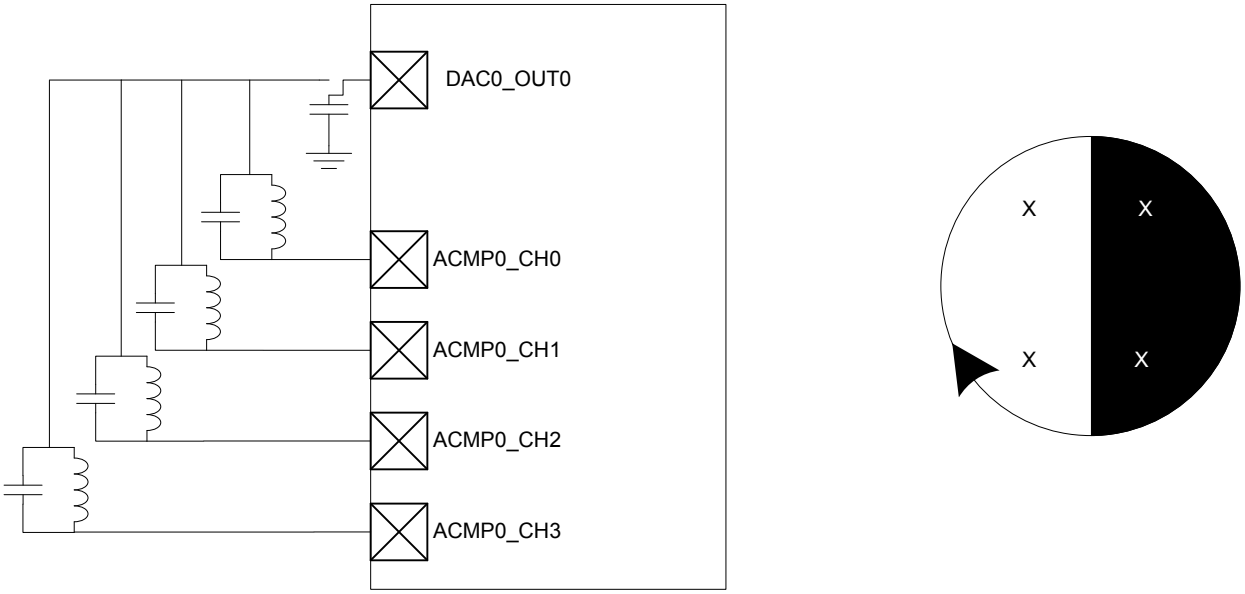


Figure 30.16. LC Sensor Setup

LESENSE can be used to excite and measure the damping factor in LC sensor oscillations. To measure the damping factor, the ACMP can be used to generate a high output each time the sensor voltage exceeds a certain level. These pulses are counted using an asynchronous counter and compared with the threshold in COMPTHRES in the CHx_EVAL register. If the number of pulses exceeds the threshold level, the sensor is said to be active, otherwise it is inactive. Figure 30.17 LC Sensor Oscillations on page 1114 illustrates how the output pulses from the ACMP correspond to damping of the oscillations. The results from sensor evaluation can automatically be fed into the decoder in order to keep track of rotations.

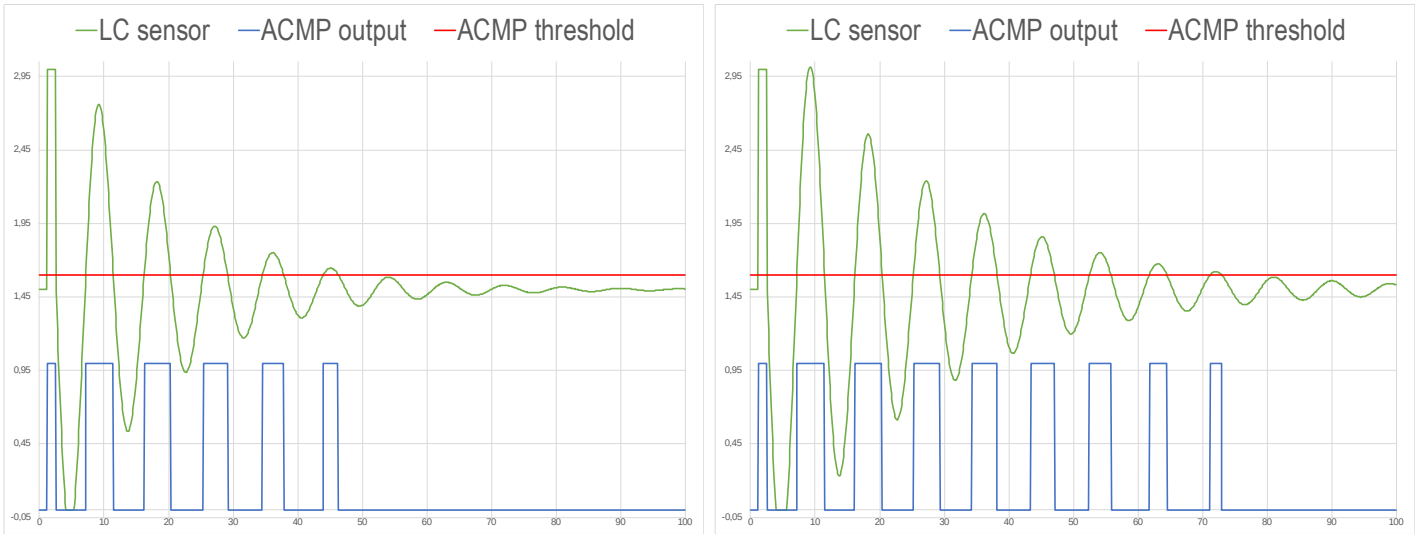


Figure 30.17. LC Sensor Oscillations

The following steps show how to configure LESENSE to scan through the four LC sensors 100 times per second.

1. Assuming $LFACLK_{LESENSE}$ is 32kHz, set PCPRESC to 3 and PCTOP to 39 in CTRL. This will set the LESENSE scan frequency to 100Hz.
2. Enable the VDAC and configure it to produce a voltage of Vdd/2.

3. Enable channels 0 through 3 in CHEN. Set IDLECONF for the active channels to DACOUT. The channel pins should be connected to the VDAC output (effectively shorting the LC sensor) in the idle phase to damp the oscillations.
4. Configure the ACMP to use scaled Vdd as negative input, refer to ACMP chapter for details.
5. Enable and configure PCNT and asynchronous PRS.
6. Configure the GPIOs used as PUSHPULL.
7. Configure the following bit fields in CHx_CONF, for channels 0 through 3:
   a. Set EXCLK to AUXHFRCO. AUXHFRCO is needed to achieve short excitation time.
   b. Set EXTIME to an appropriate value. Excitation will last for EXTIME/$F_{AUXHFRCO}$ seconds.
   c. Set EXMODE to HIGH. The LC sensors are excited by pulling the excitation pin high.
   d. Set SAMPLE to ACMPCOUNT and COMP to LESS. Status of each sensor is evaluated based on the number of pulses generated by the ACMP. If they are less than the threshold value, the sensor is said to be active.
   e. Set SAMPLEDLY to an appropriate value, each sensor will be measured for SAMPLEDLY/$F_{LFACLK\_LESENSE}$ seconds.
8. Set CTRTHRESHOLD to an appropriate value. If the sensor is active, the counter value after the measurement phase should be less than the threshold. If it is inactive, the counter value should be greater than the threshold.
9. Start scan sequence by writing a 1 to START in CMD.

Note: Exciting the LC sensor by pulling the excitation pin high allows the ESD protection in the pads to clamp any voltage swings below the ground voltage, giving a consistent starting point for the oscillations.

30.3.16.3 LESENSE Decoder 1

The example below illustrates how the LESENSE module can be used for decoding using three sensors.

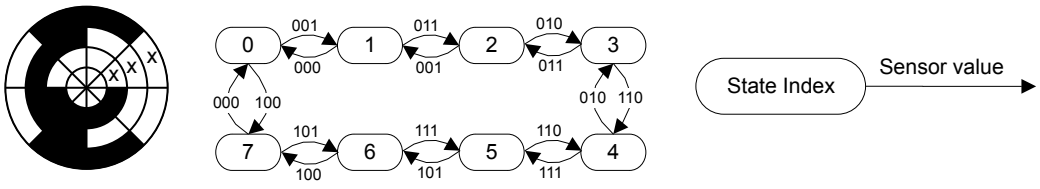


Figure 30.18. FSM Example 1

Figure 30.18 FSM Example 1 on page 1116, configure the following LESENSE registers:

1. Configure the channels to be used, be sure to set DECODE in CHx_EVAL.
2. Set PRSCNT to enable generation of count waveforms on PRS. Also configure a PCNT to listen to the PRS channels and count accordingly.
3. Configure the following in STx_TCONFA and STx_TCONFB:
   a. Set MASK = 0b1000 in STx_TCONFA and STx_TCONFB for all used states. This enables three sensors to be evaluated by the decoder.
   b. Configure the remaining bit fields in STx_TCONFA and STx_TCONFB as described in Table 30.3 LESENSE Decoder Configuration for FSM Example 1 on page 1116.
4. To initialize the decoder, run one scan, and read the present sensor status from SENSORSTATE. Then write the index of this state to DECSTATE.
5. Write to START in CMD to start scanning of sensors and decoding.

Table 30.3. LESENSE Decoder Configuration for FSM Example 1

Register	TCONFA_ NEXTSTATE	TCONFA_COMP	TCONFA_ PRSACT	TCONFB_ NEXTSTATE	TCONFB_COMP	TCONFB_ PRSACT
ST0	1	0b001	UP	7	0b100	DOWN
ST1	2	0b011	UP	0	0b000	DOWN
ST2	3	0b010	UP	1	0b001	DOWN
ST3	4	0b110	UP	2	0b011	DOWN
ST4	5	0b111	UP	3	0b010	DOWN
ST5	6	0b101	UP	4	0b110	DOWN
ST6	7	0b100	UP	5	0b111	DOWN
ST7	0	0b000	UP	6	0b101	DOWN

30.3.16.4 LESENSE Decoder 2

The example below illustrates how the LESENSE decoder can be used to implement the state machine seen in Figure 30.19 FSM Example 2 on page 1117.

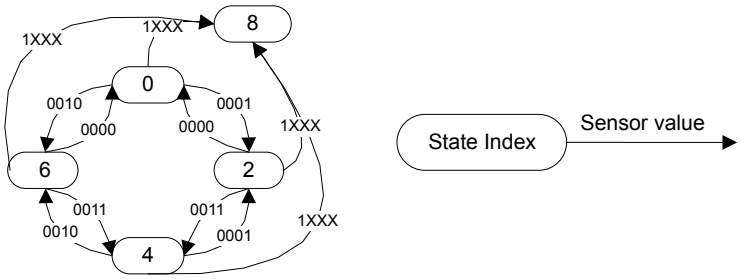


Figure 30.19. FSM Example 2

1. Configure STx_TCONFA and STx_TCONFB as described in Table 30.4 LESENSE Decoder Configuration for FSM Example 2 on page 1117.
2. To initialize the decoder, run one scan, and read the present sensor status from SENSORSTATE. Then write the index of this state to DECSTATE.
3. Write to START in CMD to start scanning of sensors and decoding.

Table 30.4. LESENSE Decoder Configuration for FSM Example 2

Register	NEXTSTATE	COMP	MASK	CHAIN
ST0_TCONFA	8	0b1000	0b0111	1
ST0_TCONFB	2	0b0001	0b1000	-
ST1_TCONFA	6	0b0010	0b1000	0
ST1_TCONFB	6	0b0010	0b1000	-
ST2_TCONFA	8	0b1000	0b0111	1
ST2_TCONFB	4	0b0011	0b1000	-
ST3_TCONFA	0	0b0000	0b1000	0
ST3_TCONFB	0	0b0000	0b1000	-
ST4_TCONFA	8	0b1000	0b0111	1
ST4_TCONFB	6	0b0010	0b1000	-
ST5_TCONFA	2	0b0001	0b1000	0
ST5_TCONFB	2	0b0001	0b1000	-
ST6_TCONFA	8	0b1000	0b0111	1
ST6_TCONFB	0	0b0000	0b1000	-
ST7_TCONFA	4	0b0011	0b1000	0
ST7_TCONFB	4	0b0011	0b1000	-

30.4 Register Map

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

Offset	Name	Type	Description
0x000	LESENSE_CTRL	RW	Control Register
0x004	LESENSE_TIMCTRL	RW	Timing Control Register
0x008	LESENSE_PERCTRL	RW	Peripheral Control Register
0x00C	LESENSE_DECCTRL	RW	Decoder Control Register
0x010	LESENSE_BIASCTRL	RW	Bias Control Register
0x014	LESENSE_EVALCTRL	RW	LESENSE Evaluation Control
0x018	LESENSE_PRSCTRL	RW	PRS Control Register
0x01C	LESENSE_CMD	W1	Command Register
0x020	LESENSE_CHEN	RW	Channel Enable Register
0x024	LESENSE_SCANRES	RWH	Scan Result Register
0x028	LESENSE_STATUS	R	Status Register
0x02C	LESENSE_PTR	R	Result Buffer Pointers
0x030	LESENSE_BUFDATA	R(a)	Result Buffer Data Register
0x034	LESENSE_CURCH	R	Current Channel Index
0x038	LESENSE_DECSTATE	RWH	Current Decoder State
0x03C	LESENSE_SENSORSTATE	RWH	Decoder Input Register
0x040	LESENSE_IDLECONF	RW	GPIO Idle Phase Configuration
0x044	LESENSE_ALTEXCONF	RW	Alternative Excite Pin Configuration
0x050	LESENSE_IF	R	Interrupt Flag Register
0x054	LESENSE_IFS	W1	Interrupt Flag Set Register
0x058	LESENSE_IFC	(R)W1	Interrupt Flag Clear Register
0x05C	LESENSE_IEN	RW	Interrupt Enable Register
0x060	LESENSE_SYNCBUSY	R	Synchronization Busy Register
0x064	LESENSE_ROUTEPEN	RW	I/O Routing Register
0x100	LESENSE_ST0_TCONFA	RW	State Transition Configuration a
0x104	LESENSE_ST0_TCONFB	RW	State Transition Configuration B
...	LESENSE_STx_TCONFA	RW	State Transition Configuration a
...	LESENSE_STx_TCONFB	RW	State Transition Configuration B
0x1F8	LESENSE_ST31_TCONFA	RW	State Transition Configuration a
0x1FC	LESENSE_ST31_TCONFB	RW	State Transition Configuration B
0x200	LESENSE_BUF0_DATA	RWH	Scan Results
...	LESENSE_BUFx_DATA	RWH	Scan Results
0x23C	LESENSE_BUF15_DATA	RWH	Scan Results
0x240	LESENSE_CH0_TIMING	RW	Scan Configuration
0x244	LESENSE_CH0_INTERACT	RW	Scan Configuration

Offset	Name	Type	Description
0x248	LESENSE_CH0_EVAL	RWH	Scan Configuration
...	LESENSE_CHx_TIMING	RW	Scan Configuration
...	LESENSE_CHx_INTERACT	RW	Scan Configuration
...	LESENSE_CHx_EVAL	RWH	Scan Configuration
0x330	LESENSE_CH15_TIMING	RW	Scan Configuration
0x334	LESENSE_CH15_INTERACT	RW	Scan Configuration
0x338	LESENSE_CH15_EVAL	RWH	Scan Configuration

30.5 Register Description

30.5.1 LESENSE_CTRL - Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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										0	0x0		0		0	0			0		0			0x0			0x0				0x0	
Access										RW	RW		RW		RW	RW			RW		RW			RW			RW				RW	
Name										DEBUGRUN	DMAWU		BUFIDL		STRSCANRES	BUFOW			DUALSAMPLE		ALTEXMAP			SCANCONF			PRSSEL				SCANMODE	

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	DEBUGRUN	0	RW	**Debug Mode Run Enable**
	Set to keep LESENSE running in debug mode.			
	Value			Description
	0			LESENSE can not start new scans in debug mode
	1			LESENSE can start new scans in debug mode
21:20	DMAWU	0x0	RW	**DMA Wake-up From EM2**
	Set buffer threshold for waking up the DMA controller when the system is in EM2			
	Value	Mode		Description
	0	DISABLE		No DMA wake-up from EM2
	1	BUFDATAV		DMA wake-up from EM2 when data is valid in the result buffer
	2	BUFLEVEL		DMA wake-up from EM2 when the result buffer is full/half-full depending on BUFIDL configuration
19	BUFIDL	0	RW	**Result Buffer Interrupt and DMA Trigger Level**
	Set buffer threshold for DMA requests and interrupt generation			
	Value	Mode		Description
	0	HALFFULL		DMA and interrupt flags set when result buffer is half-full
	1	FULL		DMA and interrupt flags set when result buffer is full
18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	STRSCANRES	0	RW	**Enable Storing of SCANRES**
	When set, SCANRES will be stored in the result buffer after each scan			

Bit	Name	Reset	Access	Description
16	BUFOW	0	RW	**Result Buffer Overwrite**
	If set, LESENSE will always write to the result buffer, even if it is full			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	DUALSAMPLE	0	RW	**Enable Dual Sample Mode**
	When set, both ACMPs will be sampled simultaneously.			
12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	ALTEXMAP	0	RW	**Alternative Excitation Map**
	This bit is used to configure which pins alternate excitation is mapped to.			

Value	Mode	Description
0	ALTEX	Alternative excitation is mapped to the LES_ALTEX pins.
1	CH	Alternative excitation is mapped to the pin of LESENSE channel (X+8 mod 16), X being the active channel.

10:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8:7	SCANCONF	0x0	RW	**Select Scan Configuration**
	These bits control which CHx_CONF registers to be used.			

Value	Mode	Description
0	DIRMAP	The channel configuration register registers used are directly mapped to the channel number.
1	INVMAP	The channel configuration register registers used are CH_{X+8}_CONF for channels 0-7 and CH_{X-8}_CONF for channels 8-15.
2	TOGGLE	The channel configuration register registers used toggles between CH_X_CONF and CH_{X+8}_CONF when channel x triggers
3	DECDEF	The decoder state defines the CONF registers to be used.

6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:2	PRSSEL	0x0	RW	**Scan Start PRS Select**
	Select PRS source for scan start if SCANMODE is set to PRS.			

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input

Bit	Name	Reset	Access	Description
	8	PRSCH8		PRS Channel 8 selected as input
	9	PRSCH9		PRS Channel 9 selected as input
	10	PRSCH10		PRS Channel 10 selected as input
	11	PRSCH11		PRS Channel 11 selected as input
	12	PRSCH12		PRS Channel 12 selected as input
	13	PRSCH13		PRS Channel 13 selected as input
	14	PRSCH14		PRS Channel 14 selected as input
	15	PRSCH15		PRS Channel 15 selected as input
1:0	SCANMODE	0x0	RW	**Configure Scan Mode**

These bits control how the scan frequency is decided

Value	Mode	Description
0	PERIODIC	A new scan is started each time the period counter overflows
1	ONESHOT	A single scan is performed when START in CMD is set
2	PRS	Pulse on PRS channel

30.5.2 LESENSE_TIMCTRL - Timing Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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				0					0x0				0x00									0x0				0x0					0x0	
Access				RW					RW				RW									RW				RW					RW	
Name				AUXSTARTUP					STARTDLY				PCTOP									PCPRESC				LFPRESC					AUXPRESC	

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	AUXSTARTUP	0	RW	**AUXHFRCO Startup Configuration**
	This bit can be set to ONDEMAND to delay startup of the AUXHFRCO when high frequency timer is used			

Value	Mode	Description
0	PREDEMAND	AUXHFRCO is started half a clock cycle before it's needed
1	ONDEMAND	AUXHFRCO is started at the time it is needed

27:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:22	STARTDLY	0x0	RW	**Start Delay Configuration**
	Delay sensor interaction STARTDELAY $LFACLK_{LESENSE}$ cycles for each channel			
21:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:12	PCTOP	0x00	RW	**Period Counter Top Value**
	These bits contain the top value for the period counter.			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	PCPRESC	0x0	RW	**Period Counter Prescaling**
	This bitfield is used to divide the clock to the period counter			

Value	Mode	Description
0	DIV1	The period counter clock frequency is $LFACLK_{LESENSE}$/1
1	DIV2	The period counter clock frequency is $LFACLK_{LESENSE}$/2
2	DIV4	The period counter clock frequency is $LFACLK_{LESENSE}$/4
3	DIV8	The period counter clock frequency is $LFACLK_{LESENSE}$/8
4	DIV16	The period counter clock frequency is $LFACLK_{LESENSE}$/16
5	DIV32	The period counter clock frequency is $LFACLK_{LESENSE}$/32

Bit	Name	Reset	Access	Description
	6	DIV64		The period counter clock frequency is $LFACLK_{LESENSE}$/64
	7	DIV128		The period counter clock frequency is $LFACLK_{LESENSE}$/128
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	LFPRESC	0x0	RW	**Prescaling Factor for Low Frequency Timer**
	This bitfield is used to divide the clock to the low frequency timer			
	Value	Mode		Description
	0	DIV1		Low frequency timer is clocked with $LFACLK_{LESENSE}$/1
	1	DIV2		Low frequency timer is clocked with $LFACLK_{LESENSE}$/2
	2	DIV4		Low frequency timer is clocked with $LFACLK_{LESENSE}$/4
	3	DIV8		Low frequency timer is clocked with $LFACLK_{LESENSE}$/8
	4	DIV16		Low frequency timer is clocked with $LFACLK_{LESENSE}$/16
	5	DIV32		Low frequency timer is clocked with $LFACLK_{LESENSE}$/32
	6	DIV64		Low frequency timer is clocked with $LFACLK_{LESENSE}$/64
	7	DIV128		Low frequency timer is clocked with $LFACLK_{LESENSE}$/128
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	AUXPRESC	0x0	RW	**Prescaling Factor for High Frequency Timer**
	This bitfield is used to divide the clock to the high frequency timer			
	Value	Mode		Description
	0	DIV1		High frequency timer is clocked with AUXHFRCO/1
	1	DIV2		High frequency timer is clocked with AUXHFRCO/2
	2	DIV4		High frequency timer is clocked with AUXHFRCO/4
	3	DIV8		High frequency timer is clocked with AUXHFRCO/8

30.5.3 LESENSE_PERCTRL - Peripheral Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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		0	0	0	0	0x0		0x0													0		0			0	0	0	0
Access			RW		RW	RW	RW	RW	RW		RW													RW		RW			RW	RW	RW	RW
Name			WARMUPMODE		ACMP1HYSTEN	ACMP0HYSTEN	ACMP1INV	ACMP0INV	ACMP1MODE		ACMP0MODE													DACCONVTRIG		DACSTARTUP			DACCH1DATA	DACCH0DATA	DACCH1EN	DACCH0EN

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	WARMUPMODE	0x0	RW	**ACMP and VDAC Duty Cycle Mode**

This bitfield is used to configure how the VDAC and ACMP are duty cycled when LESENSE is controlling them

Value	Mode	Description
0	NORMAL	The analog comparators and VDAC are shut down when LESENSE is idle
1	KEEPACMPWARM	The analog comparators are kept powered up when LESENSE is idle
2	KEEPDACWARM	The VDAC is kept powered up when LESENSE is idle
3	KEEPACMPDACWARM	The analog comparators and VDAC are kept powered up when LESENSE is idle

27	ACMP1HYSTEN	0	RW	**ACMP1 Hysteresis Enable**

Set to control ACMP1_HYSTERESIS0_DIVVX and ACMP1_HYSTERESIS1_DIVVX separately.

26	ACMP0HYSTEN	0	RW	**ACMP0 Hysteresis Enable**

Set to control ACMP0_HYSTERESIS0_DIVVX and ACMP0_HYSTERESIS1_DIVVX separately.

25	ACMP1INV	0	RW	**Invert Analog Comparator 1 Output**

This bit can be set to invert the output coming from ACMP1

24	ACMP0INV	0	RW	**Invert Analog Comparator 0 Output**

This bit can be set to invert the output coming from ACMP0

23:22	ACMP1MODE	0x0	RW	**ACMP1 Mode**

Configure how LESENSE controls ACMP1

Value	Mode	Description
0	DISABLE	LESENSE does not control ACMP1
1	MUX	LESENSE controls the input mux (POSSEL) of ACMP1
2	MUXTHRES	LESENSE controls the input mux and the threshold value (VDDLEVEL) of ACMP1

Bit	Name	Reset	Access	Description
21:20	ACMP0MODE	0x0	RW	**ACMP0 Mode**

Configure how LESENSE controls ACMP0

Value	Mode	Description
0	DISABLE	LESENSE does not control ACMP0
1	MUX	LESENSE controls the input mux (POSSEL) of ACMP0
2	MUXTHRES	LESENSE controls the input mux (POSSEL) and the threshold value (VDDLEVEL) of ACMP0

19:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	DACCONVTRIG	0	RW	**VDAC Conversion Trigger Configuration**

This bit is used to configure how frequently a VDAC conversion is triggered

Value	Mode	Description
0	CHANNELSTART	VDAC is enabled before every LESENSE channel measurement.
1	SCANSTART	VDAC is only enabled once per scan.

7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	DACSTARTUP	0	RW	**VDAC Startup Configuration**

This bit is used to configure the duration between the VDAC conversion trigger and the sensor interaction

Value	Mode	Description
0	FULLCYCLE	VDAC is started a full $LFACLK_{LESENSE}$ cycle before sensor interaction starts.
1	HALFCYCLE	VDAC is started half a $LFACLK_{LESENSE}$ cycle before sensor interaction starts.

5:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	DACCH1DATA	0	RW	**VDAC CH1 Data Selection**

This bit decides if the data used for VDAC conversion is taken from the VDAC interface or from LESENSE

Value	Mode	Description
0	DACDATA	VDAC data is defined by CH1DATA in the VDAC interface.
1	THRES	VDAC data is defined by THRES in CHx_INTERACT.

2	DACCH0DATA	0	RW	**VDAC CH0 Data Selection**

This bit decides if the data used for VDAC conversion is taken from the VDAC interface or from LESENSE

Value	Mode	Description
0	DACDATA	VDAC data is defined by CH0DATA in the VDAC interface.
1	THRES	VDAC data is defined by THRES in CHx_INTERACT.

Bit	Name	Reset	Access	Description
1	DACCH1EN	0	RW	**VDAC CH1 Enable**
	Enable LESENSE control of VDAC0 CH1			
0	DACCH0EN	0	RW	**VDAC CH0 Enable**
	Enable LESENSE control of VDAC0 CH0			

30.5.4 LESENSE_DECCTRL - Decoder Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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					0	0	0	0	0	0	0	0	0
Access				RW					RW					RW					RW					RW	RW	RW	RW	RW	RW	RW	RW	RW
Name				PRSSEL3					PRSSEL2					PRSSEL1					PRSSEL0					INPUT	PRSCNT	HYSTIRQ	HYSTPRS2	HYSTPRS1	HYSTPRS0	INTMAP	ERRCHK	DISABLE

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*		
28:25	PRSSEL3	0x0	RW	**LESENSE Decoder PRS Input 3 Configuration**

Select PRS input for bit 3 of the LESENSE decoder

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input
8	PRSCH8	PRS Channel 8 selected as input
9	PRSCH9	PRS Channel 9 selected as input
10	PRSCH10	PRS Channel 10 selected as input
11	PRSCH11	PRS Channel 11 selected as input
12	PRSCH12	PRS Channel 12 selected as input
13	PRSCH13	PRS Channel 13 selected as input
14	PRSCH14	PRS Channel 14 selected as input
15	PRSCH15	PRS Channel 15 selected as input

24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23:20	PRSSEL2	0x0	RW	**LESENSE Decoder PRS Input 2 Configuration**

Select PRS input for bit 2 of the LESENSE decoder

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input

Bit	Name	Reset	Access	Description
	1	PRSCH1		PRS Channel 1 selected as input
	2	PRSCH2		PRS Channel 2 selected as input
	3	PRSCH3		PRS Channel 3 selected as input
	4	PRSCH4		PRS Channel 4 selected as input
	5	PRSCH5		PRS Channel 5 selected as input
	6	PRSCH6		PRS Channel 6 selected as input
	7	PRSCH7		PRS Channel 7 selected as input
	8	PRSCH8		PRS Channel 8 selected as input
	9	PRSCH9		PRS Channel 9 selected as input
	10	PRSCH10		PRS Channel 10 selected as input
	11	PRSCH11		PRS Channel 11 selected as input
	12	PRSCH12		PRS Channel 12 selected as input
	13	PRSCH13		PRS Channel 13 selected as input
	14	PRSCH14		PRS Channel 14 selected as input
	15	PRSCH15		PRS Channel 15 selected as input
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:15	PRSSEL1	0x0	RW	**LESENSE Decoder PRS Input 1 Configuration**
	Select PRS input for the bit 1 of the LESENSE decoder			
	Value	Mode		Description
	0	PRSCH0		PRS Channel 0 selected as input
	1	PRSCH1		PRS Channel 1 selected as input
	2	PRSCH2		PRS Channel 2 selected as input
	3	PRSCH3		PRS Channel 3 selected as input
	4	PRSCH4		PRS Channel 4 selected as input
	5	PRSCH5		PRS Channel 5 selected as input
	6	PRSCH6		PRS Channel 6 selected as input
	7	PRSCH7		PRS Channel 7 selected as input
	8	PRSCH8		PRS Channel 8 selected as input
	9	PRSCH9		PRS Channel 9 selected as input
	10	PRSCH10		PRS Channel 10 selected as input
	11	PRSCH11		PRS Channel 11 selected as input
	12	PRSCH12		PRS Channel 12 selected as input
	13	PRSCH13		PRS Channel 13 selected as input
	14	PRSCH14		PRS Channel 14 selected as input
	15	PRSCH15		PRS Channel 15 selected as input

Bit	Name	Reset	Access	Description
14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:10	PRSSEL0	0x0	RW	**LESENSE Decoder PRS Input 0 Configuration**

Select PRS input for the bit 0 of the LESENSE decoder

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as input
1	PRSCH1	PRS Channel 1 selected as input
2	PRSCH2	PRS Channel 2 selected as input
3	PRSCH3	PRS Channel 3 selected as input
4	PRSCH4	PRS Channel 4 selected as input
5	PRSCH5	PRS Channel 5 selected as input
6	PRSCH6	PRS Channel 6 selected as input
7	PRSCH7	PRS Channel 7 selected as input
8	PRSCH8	PRS Channel 8 selected as input
9	PRSCH9	PRS Channel 9 selected as input
10	PRSCH10	PRS Channel 10 selected as input
11	PRSCH11	PRS Channel 11 selected as input
12	PRSCH12	PRS Channel 12 selected as input
13	PRSCH13	PRS Channel 13 selected as input
14	PRSCH14	PRS Channel 14 selected as input
15	PRSCH15	PRS Channel 15 selected as input

9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	INPUT	0	RW	**LESENSE Decoder Input Configuration**

Select input to the LESENSE decoder

Value	Mode	Description
0	SENSORSTATE	The SENSORSTATE register is used as input to the decoder.
1	PRS	PRS channels are used as input to the decoder.

7	PRSCNT	0	RW	**Enable Count Mode on Decoder PRS Channels 0 and 1**

When set, decoder PRS0 and PRS1 will be used to produce output which can be used by a PCNT to count up or down.

6	HYSTIRQ	0	RW	**Enable Decoder Hysteresis on Interrupt Requests**

When set, hysteresis is enabled in the decoder, suppressing interrupt requests.

5	HYSTPRS2	0	RW	**Enable Decoder Hysteresis on PRS2 Output**

When set, hysteresis is enabled in the decoder, suppressing changes on PRS channel 2

4	HYSTPRS1	0	RW	**Enable Decoder Hysteresis on PRS1 Output**

When set, hysteresis is enabled in the decoder, suppressing changes on PRS channel 1

Bit	Name	Reset	Access	Description
3	HYSTPRS0	0	RW	**Enable Decoder Hysteresis on PRS0 Output**
	When set, hysteresis is enabled in the decoder, suppressing changes on PRS channel 0			
2	INTMAP	0	RW	**Enable Decoder to Channel Interrupt Mapping**
	When set, a transition from state x in the decoder will set interrupt flag CH[x mod 16]			
1	ERRCHK	0	RW	**Enable Check of Current State**
	When set, the decoder checks the current state in addition to the states defined in TCONF			
0	DISABLE	0	RW	**Disable the Decoder**
	When set, the decoder is disabled. When disabled the decoder will keep its current state			

30.5.5 LESENSE_BIASCTRL - Bias Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																															BIASMODE	

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	BIASMODE	0x0	RW	**Select Bias Mode**
	This bitfield is used to configure how LESENSE interacts with the bias module			

Value	Mode	Description
0	DONTTOUCH	Bias module is controlled by the EMU and is not affected by LESENSE
1	DUTYCYCLE	Bias module duty cycled between low power and high accuracy mode
2	HIGHACC	Bias module always in high accuracy mode

30.5.6 LESENSE_EVALCTRL - LESENSE Evaluation Control (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																	0x0000															
Access																	RW															
Name																	WINSIZE															

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	WINSIZE	0x0000	RW	**Sliding Window and Step Detection Size** In sliding window mode, this bitfield configures the window size. In step detection mode, this bitfield is used to configure the threshold for step detection

30.5.7 LESENSE_PRSCTRL - PRS Control Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																0				0x00								0x00				
Access																RW				RW								RW				
Name																DECCMPEN				DECCMPMASK								DECCMPVAL				

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	DECCMPEN	0	RW	**Enable PRS Output DECCMP**
	Enables decoder state compare match PRS output			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:8	DECCMPMASK	0x00	RW	**Decoder State Compare Value Mask**
	Masks DECCMPVAL and DECSTATE for comparison			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4:0	DECCMPVAL	0x00	RW	**Decoder State Compare Value**
	Triggers PRS output when equal to DECSTATE			

30.5.8 LESENSE_CMD - Command Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																													0	0	0	0
Access																													W1	W1	W1	W1
Name																													CLEARBUF	DECODE	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	CLEARBUF	0	W1	**Clear Result Buffer**
	Set this bit to reset the read and write pointers of the result buffer.			
2	DECODE	0	W1	**Start Decoder**
	Set this bit to start the LESENSE decoder.			
1	STOP	0	W1	**Stop Scanning of Sensors**
	Set this bit to stop LESENSE. If issued during a scan, the command will take effect after scan completion.			
0	START	0	W1	**Start Scanning of Sensors**
	Set this bit to start LESENSE.			

30.5.9 LESENSE_CHEN - Channel Enable Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																	0x0000															
Access																	RW															
Name																	CHEN															

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	CHEN	0x0000	RW	**Enable Scan Channel**
	Set bit X to enable channel X			

30.5.10 LESENSE_SCANRES - Scan Result Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	0x0000																0x0000															
Access	RWH																RWH															
Name	STEPDIR																SCANRES															

Bit	Name	Reset	Access	Description
31:16	STEPDIR	0x0000	RWH	**Direction of Previous Step Detection**
	In step detection mode, bit X will be set if a step up was detected on channel X			
15:0	SCANRES	0x0000	RWH	**Scan Results**
	Bit X will be set depending on channel X evaluation			

30.5.11 LESENSE_STATUS - Status Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																											0	0	0	0	0	0
Access																											R	R	R	R	R	R
Name																											DACACTIVE	SCANACTIVE	RUNNING	BUFFULL	BUFHALFFULL	BUFDATAV

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	DACACTIVE	0	R	**LESENSE VDAC Interface is Active**
	LESENSE is currently using the VDAC.			
4	SCANACTIVE	0	R	**LESENSE Scan Active**
	LESENSE is currently interfacing to sensors.			
3	RUNNING	0	R	**LESENSE Periodic Counter Running**
	LESENSE is running in periodic mode.			
2	BUFFULL	0	R	**Result Buffer Full**
	Set when the result buffer is full			
1	BUFHALFFULL	0	R	**Result Buffer Half Full**
	Set when the result buffer is half full			
0	BUFDATAV	0	R	**Result Data Valid**
	Set when data is available in the result buffer. Cleared when the buffer is empty.			

30.5.12 LESENSE_PTR - Result Buffer Pointers (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																									WR				RD			

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	WR	0x0	R	**Result Buffer Write Pointer**
	These bits show the next index in the result buffer to be written to. Incremented when LESENSE writes to result buffer			
3:0	RD	0x0	R	**Result Buffer Read Pointer**
	These bits show the index of the oldest unread data in the result buffer. Incremented on read from BUFDATA.			

30.5.13 LESENSE_BUFDATA - Result Buffer Data Register (Async Reg) (Actionable Reads)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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													0xX				0xXXXX															
Access													R				R															
Name													BUFDATASRC				BUFDATA															

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	BUFDATASRC	0xX	R	**Result Data Source**
	This bitfield contains the channel index for the sensor result in BUFDATA.			
15:0	BUFDATA	0xXXXX	R	**Result Data**
	This register can be used to read the oldest unread data from the result buffer.			

30.5.14 LESENSE_CURCH - Current Channel Index (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																													R		
Name																													CURCH		

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	CURCH	0x0	R	**Current Channel Index**
	Shows the index of the current channel			

30.5.15 LESENSE_DECSTATE - Current Decoder State (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																														0x00		
Access																														RWH		
Name																														DECSTATE		

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	DECSTATE	0x00	RWH	**Current Decoder State**
	Shows the current decoder state			

30.5.16 LESENSE_SENSORSTATE - Decoder Input Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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																													RWH			
Name																													SENSORSTATE			

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	SENSORSTATE	0x0	RWH	**Decoder Input Register**
	Shows the status of sensors chosen as input to the decoder			

30.5.17 LESENSE_IDLECONF - GPIO Idle Phase Configuration (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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	
Access	RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW	
Name	CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8		CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	

Bit	Name	Reset	Access	Description
31:30	CH15	0x0	RW	**Channel 15 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description
0	DISABLE	CH15 output is disabled in idle phase
1	HIGH	CH15 output is high in idle phase
2	LOW	CH15 output is low in idle phase
3	DAC	CH15 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

Bit	Name	Reset	Access	Description
29:28	CH14	0x0	RW	**Channel 14 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description
0	DISABLE	CH14 output is disabled in idle phase
1	HIGH	CH14 output is high in idle phase
2	LOW	CH14 output is low in idle phase
3	DAC	CH14 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

Bit	Name	Reset	Access	Description
27:26	CH13	0x0	RW	**Channel 13 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description
0	DISABLE	CH13 output is disabled in idle phase
1	HIGH	CH13 output is high in idle phase
2	LOW	CH13 output is low in idle phase
3	DAC	CH13 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

Bit	Name	Reset	Access	Description
25:24	CH12	0x0	RW	**Channel 12 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	DISABLE		CH12 output is disabled in idle phase
	1	HIGH		CH12 output is high in idle phase
	2	LOW		CH12 output is low in idle phase
	3	DAC		CH12 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15
23:22	CH11	0x0	RW	**Channel 11 Idle Phase Configuration**
	This bitfield determines how the channel is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		CH11 output is disabled in idle phase
	1	HIGH		CH11 output is high in idle phase
	2	LOW		CH11 output is low in idle phase
	3	DAC		CH11 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15
21:20	CH10	0x0	RW	**Channel 10 Idle Phase Configuration**
	This bitfield determines how the channel is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		CH10 output is disabled in idle phase
	1	HIGH		CH10 output is high in idle phase
	2	LOW		CH10 output is low in idle phase
	3	DAC		CH10 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15
19:18	CH9	0x0	RW	**Channel 9 Idle Phase Configuration**
	This bitfield determines how the channel is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		CH9 output is disabled in idle phase
	1	HIGH		CH9 output is high in idle phase
	2	LOW		CH9 output is low in idle phase
	3	DAC		CH9 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15
17:16	CH8	0x0	RW	**Channel 8 Idle Phase Configuration**
	This bitfield determines how the channel is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		CH8 output is disabled in idle phase
	1	HIGH		CH8 output is high in idle phase
	2	LOW		CH8 output is low in idle phase
	3	DAC		CH8 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

Bit	Name	Reset	Access	Description
15:14	CH7	0x0	RW	**Channel 7 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description
0	DISABLE	CH7 output is disabled in idle phase
1	HIGH	CH7 output is high in idle phase
2	LOW	CH7 output is low in idle phase
3	DAC	CH7 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

Bit	Name	Reset	Access	Description
13:12	CH6	0x0	RW	**Channel 6 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description
0	DISABLE	CH6 output is disabled in idle phase
1	HIGH	CH6 output is high in idle phase
2	LOW	CH6 output is low in idle phase
3	DAC	CH6 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

Bit	Name	Reset	Access	Description
11:10	CH5	0x0	RW	**Channel 5 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description
0	DISABLE	CH5 output is disabled in idle phase
1	HIGH	CH5 output is high in idle phase
2	LOW	CH5 output is low in idle phase
3	DAC	CH5 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

Bit	Name	Reset	Access	Description
9:8	CH4	0x0	RW	**Channel 4 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description
0	DISABLE	CH4 output is disabled in idle phase
1	HIGH	CH4 output is high in idle phase
2	LOW	CH4 output is low in idle phase
3	DAC	CH4 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

Bit	Name	Reset	Access	Description
7:6	CH3	0x0	RW	**Channel 3 Idle Phase Configuration**

This bitfield determines how the channel is configured during the idle phase

Value	Mode	Description
0	DISABLE	CH3 output is disabled in idle phase

Bit	Name	Reset	Access	Description
	1	HIGH		CH3 output is high in idle phase
	2	LOW		CH3 output is low in idle phase
	3	DAC		CH3 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15
5:4	CH2	0x0	RW	**Channel 2 Idle Phase Configuration**
	This bitfield determines how the channel is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		CH2 output is disabled in idle phase
	1	HIGH		CH2 output is high in idle phase
	2	LOW		CH2 output is low in idle phase
	3	DAC		CH2 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15
3:2	CH1	0x0	RW	**Channel 1 Idle Phase Configuration**
	This bitfield determines how the channel is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		CH1 output is disabled in idle phase
	1	HIGH		CH1 output is high in idle phase
	2	LOW		CH1 output is low in idle phase
	3	DAC		CH1 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15
1:0	CH0	0x0	RW	**Channel 0 Idle Phase Configuration**
	This bitfield determines how the channel is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		CH0 output is disabled in idle phase
	1	HIGH		CH0 output is high in idle phase
	2	LOW		CH0 output is low in idle phase
	3	DAC		CH0 output is connected to VDAC output in idle phase. Note that this mode is only available on channels 0, 1, 2, 3, 12, 13, 14, 15

30.5.18 LESENSE_ALTEXCONF - Alternative Excite Pin Configuration (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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									0	0	0	0	0	0	0	0	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									AEX7	AEX6	AEX5	AEX4	AEX3	AEX2	AEX1	AEX0	IDLECONF7		IDLECONF6		IDLECONF5		IDLECONF4		IDLECONF3		IDLECONF2		IDLECONF1		IDLECONF0	

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	AEX7	0	RW	**ALTEX7 Always Excite Enable**
	Set this bit to excite ALTEX7 regardless of what channel is active			
22	AEX6	0	RW	**ALTEX6 Always Excite Enable**
	Set this bit to excite ALTEX6 regardless of what channel is active			
21	AEX5	0	RW	**ALTEX5 Always Excite Enable**
	Set this bit to excite ALTEX5 regardless of what channel is active			
20	AEX4	0	RW	**ALTEX4 Always Excite Enable**
	Set this bit to excite ALTEX4 regardless of what channel is active			
19	AEX3	0	RW	**ALTEX3 Always Excite Enable**
	Set this bit to excite ALTEX3 regardless of what channel is active			
18	AEX2	0	RW	**ALTEX2 Always Excite Enable**
	Set this bit to excite ALTEX2 regardless of what channel is active			
17	AEX1	0	RW	**ALTEX1 Always Excite Enable**
	Set this bit to excite ALTEX1 regardless of what channel is active			
16	AEX0	0	RW	**ALTEX0 Always Excite Enable**
	Set this bit to excite ALTEX0 regardless of what channel is active			
15:14	IDLECONF7	0x0	RW	**ALTEX7 Idle Phase Configuration**
	This bitfield determines how the alternate excite pin is configured during the idle phase			

Value	Mode	Description
0	DISABLE	ALTEX7 output is disabled in idle phase
1	HIGH	ALTEX7 output is high in idle phase
2	LOW	ALTEX7 output is low in idle phase

13:12	IDLECONF6	0x0	RW	**ALTEX6 Idle Phase Configuration**
	This bitfield determines how the alternate excite pin is configured during the idle phase			

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	DISABLE		ALTEX6 output is disabled in idle phase
	1	HIGH		ALTEX6 output is high in idle phase
	2	LOW		ALTEX6 output is low in idle phase
11:10	IDLECONF5	0x0	RW	**ALTEX5 Idle Phase Configuration**
	This bitfield determines how the alternate excite pin is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		ALTEX5 output is disabled in idle phase
	1	HIGH		ALTEX5 output is high in idle phase
	2	LOW		ALTEX5 output is low in idle phase
9:8	IDLECONF4	0x0	RW	**ALTEX4 Idle Phase Configuration**
	This bitfield determines how the alternate excite pin is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		ALTEX4 output is disabled in idle phase
	1	HIGH		ALTEX4 output is high in idle phase
	2	LOW		ALTEX4 output is low in idle phase
7:6	IDLECONF3	0x0	RW	**ALTEX3 Idle Phase Configuration**
	This bitfield determines how the alternate excite pin is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		ALTEX3 output is disabled in idle phase
	1	HIGH		ALTEX3 output is high in idle phase
	2	LOW		ALTEX3 output is low in idle phase
5:4	IDLECONF2	0x0	RW	**ALTEX2 Idle Phase Configuration**
	This bitfield determines how the alternate excite pin is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		ALTEX2 output is disabled in idle phase
	1	HIGH		ALTEX2 output is high in idle phase
	2	LOW		ALTEX2 output is low in idle phase
3:2	IDLECONF1	0x0	RW	**ALTEX1 Idle Phase Configuration**
	This bitfield determines how the alternate excite pin is configured during the idle phase			
	Value	Mode		Description
	0	DISABLE		ALTEX1 output is disabled in idle phase
	1	HIGH		ALTEX1 output is high in idle phase
	2	LOW		ALTEX1 output is low in idle phase

Bit	Name	Reset	Access	Description
1:0	IDLECONF0	0x0	RW	**ALTEX0 Idle Phase Configuration**

This bitfield determines how the alternate excite pin is configured during the idle phase

Value	Mode	Description
0	DISABLE	ALTEX0 output is disabled in idle phase
1	HIGH	ALTEX0 output is high in idle phase
2	LOW	ALTEX0 output is low in idle phase

30.5.19 LESENSE_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										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access										R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name										CNTOF	BUFOF	BUFLEVEL	BUFDATAV	DECERR	DEC	SCANCOMPLETE	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

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	CNTOF	0	R	**CNTOF Interrupt Flag**
	Set when the LESENSE counter overflows.			
21	BUFOF	0	R	**BUFOF Interrupt Flag**
	Set when the result buffer overflows			
20	BUFLEVEL	0	R	**BUFLEVEL Interrupt Flag**
	Set when the data buffer is full.			
19	BUFDATAV	0	R	**BUFDATAV Interrupt Flag**
	Set when data is available in the result buffer.			
18	DECERR	0	R	**DECERR Interrupt Flag**
	Set when the decoder detects an error			
17	DEC	0	R	**DEC Interrupt Flag**
	Set when the decoder has issued an interrupt request			
16	SCANCOMPLETE	0	R	**SCANCOMPLETE Interrupt Flag**
	Set when a scan sequence is completed			
15	CH15	0	R	**CH15 Interrupt Flag**
	Set when channel 15 triggers			
14	CH14	0	R	**CH14 Interrupt Flag**
	Set when channel 14 triggers			
13	CH13	0	R	**CH13 Interrupt Flag**
	Set when channel 13 triggers			
12	CH12	0	R	**CH12 Interrupt Flag**
	Set when channel 12 triggers			
11	CH11	0	R	**CH11 Interrupt Flag**
	Set when channel 11 triggers			
10	CH10	0	R	**CH10 Interrupt Flag**
	Set when channel 10 triggers			

Bit	Name	Reset	Access	Description
9	CH9	0	R	**CH9 Interrupt Flag**
	Set when channel 9 triggers			
8	CH8	0	R	**CH8 Interrupt Flag**
	Set when channel 8 triggers			
7	CH7	0	R	**CH7 Interrupt Flag**
	Set when channel 7 triggers			
6	CH6	0	R	**CH6 Interrupt Flag**
	Set when channel 6 triggers			
5	CH5	0	R	**CH5 Interrupt Flag**
	Set when channel 5 triggers			
4	CH4	0	R	**CH4 Interrupt Flag**
	Set when channel 4 triggers			
3	CH3	0	R	**CH3 Interrupt Flag**
	Set when channel 3 triggers			
2	CH2	0	R	**CH2 Interrupt Flag**
	Set when channel 2 triggers			
1	CH1	0	R	**CH1 Interrupt Flag**
	Set when channel 1 triggers			
0	CH0	0	R	**CH0 Interrupt Flag**
	Set when channel 0 triggers			

30.5.20 LESENSE_IFS - Interrupt Flag Set 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										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access										W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name										CNTOF	BUFOF	BUFLEVEL	BUFDATAV	DECERR	DEC	SCANCOMPLETE	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

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	CNTOF	0	W1	**Set CNTOF Interrupt Flag**
	Write 1 to set the CNTOF interrupt flag			
21	BUFOF	0	W1	**Set BUFOF Interrupt Flag**
	Write 1 to set the BUFOF interrupt flag			
20	BUFLEVEL	0	W1	**Set BUFLEVEL Interrupt Flag**
	Write 1 to set the BUFLEVEL interrupt flag			
19	BUFDATAV	0	W1	**Set BUFDATAV Interrupt Flag**
	Write 1 to set the BUFDATAV interrupt flag			
18	DECERR	0	W1	**Set DECERR Interrupt Flag**
	Write 1 to set the DECERR interrupt flag			
17	DEC	0	W1	**Set DEC Interrupt Flag**
	Write 1 to set the DEC interrupt flag			
16	SCANCOMPLETE	0	W1	**Set SCANCOMPLETE Interrupt Flag**
	Write 1 to set the SCANCOMPLETE interrupt flag			
15	CH15	0	W1	**Set CH15 Interrupt Flag**
	Write 1 to set the CH15 interrupt flag			
14	CH14	0	W1	**Set CH14 Interrupt Flag**
	Write 1 to set the CH14 interrupt flag			
13	CH13	0	W1	**Set CH13 Interrupt Flag**
	Write 1 to set the CH13 interrupt flag			
12	CH12	0	W1	**Set CH12 Interrupt Flag**
	Write 1 to set the CH12 interrupt flag			
11	CH11	0	W1	**Set CH11 Interrupt Flag**
	Write 1 to set the CH11 interrupt flag			

Bit	Name	Reset	Access	Description
10	CH10	0	W1	**Set CH10 Interrupt Flag**
	Write 1 to set the CH10 interrupt flag			
9	CH9	0	W1	**Set CH9 Interrupt Flag**
	Write 1 to set the CH9 interrupt flag			
8	CH8	0	W1	**Set CH8 Interrupt Flag**
	Write 1 to set the CH8 interrupt flag			
7	CH7	0	W1	**Set CH7 Interrupt Flag**
	Write 1 to set the CH7 interrupt flag			
6	CH6	0	W1	**Set CH6 Interrupt Flag**
	Write 1 to set the CH6 interrupt flag			
5	CH5	0	W1	**Set CH5 Interrupt Flag**
	Write 1 to set the CH5 interrupt flag			
4	CH4	0	W1	**Set CH4 Interrupt Flag**
	Write 1 to set the CH4 interrupt flag			
3	CH3	0	W1	**Set CH3 Interrupt Flag**
	Write 1 to set the CH3 interrupt flag			
2	CH2	0	W1	**Set CH2 Interrupt Flag**
	Write 1 to set the CH2 interrupt flag			
1	CH1	0	W1	**Set CH1 Interrupt Flag**
	Write 1 to set the CH1 interrupt flag			
0	CH0	0	W1	**Set CH0 Interrupt Flag**
	Write 1 to set the CH0 interrupt flag			

30.5.21 LESENSE_IFC - Interrupt Flag Clear 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										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access										(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name										CNTOF	BUFOF	BUFLEVEL	BUFDATAV	DECERR	DEC	SCANCOMPLETE	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

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	CNTOF	0	(R)W1	**Clear CNTOF Interrupt Flag**
	Write 1 to clear the CNTOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
21	BUFOF	0	(R)W1	**Clear BUFOF Interrupt Flag**
	Write 1 to clear the BUFOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
20	BUFLEVEL	0	(R)W1	**Clear BUFLEVEL Interrupt Flag**
	Write 1 to clear the BUFLEVEL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
19	BUFDATAV	0	(R)W1	**Clear BUFDATAV Interrupt Flag**
	Write 1 to clear the BUFDATAV interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
18	DECERR	0	(R)W1	**Clear DECERR Interrupt Flag**
	Write 1 to clear the DECERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
17	DEC	0	(R)W1	**Clear DEC Interrupt Flag**
	Write 1 to clear the DEC interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
16	SCANCOMPLETE	0	(R)W1	**Clear SCANCOMPLETE Interrupt Flag**
	Write 1 to clear the SCANCOMPLETE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
15	CH15	0	(R)W1	**Clear CH15 Interrupt Flag**
	Write 1 to clear the CH15 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
14	CH14	0	(R)W1	**Clear CH14 Interrupt Flag**
	Write 1 to clear the CH14 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

Bit	Name	Reset	Access	Description
13	CH13	0	(R)W1	**Clear CH13 Interrupt Flag**
	Write 1 to clear the CH13 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
12	CH12	0	(R)W1	**Clear CH12 Interrupt Flag**
	Write 1 to clear the CH12 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
11	CH11	0	(R)W1	**Clear CH11 Interrupt Flag**
	Write 1 to clear the CH11 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
10	CH10	0	(R)W1	**Clear CH10 Interrupt Flag**
	Write 1 to clear the CH10 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	CH9	0	(R)W1	**Clear CH9 Interrupt Flag**
	Write 1 to clear the CH9 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	CH8	0	(R)W1	**Clear CH8 Interrupt Flag**
	Write 1 to clear the CH8 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7	CH7	0	(R)W1	**Clear CH7 Interrupt Flag**
	Write 1 to clear the CH7 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
6	CH6	0	(R)W1	**Clear CH6 Interrupt Flag**
	Write 1 to clear the CH6 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
5	CH5	0	(R)W1	**Clear CH5 Interrupt Flag**
	Write 1 to clear the CH5 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
4	CH4	0	(R)W1	**Clear CH4 Interrupt Flag**
	Write 1 to clear the CH4 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	CH3	0	(R)W1	**Clear CH3 Interrupt Flag**
	Write 1 to clear the CH3 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	CH2	0	(R)W1	**Clear CH2 Interrupt Flag**
	Write 1 to clear the CH2 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	CH1	0	(R)W1	**Clear CH1 Interrupt Flag**
	Write 1 to clear the CH1 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	CH0	0	(R)W1	**Clear CH0 Interrupt Flag**
	Write 1 to clear the CH0 interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

30.5.22 LESENSE_IEN - Interrupt Enable 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										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access										RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name										CNTOF	BUFOF	BUFLEVEL	BUFDATAV	DECERR	DEC	SCANCOMPLETE	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

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	CNTOF	0	RW	**CNTOF Interrupt Enable**
	Enable/disable the CNTOF interrupt			
21	BUFOF	0	RW	**BUFOF Interrupt Enable**
	Enable/disable the BUFOF interrupt			
20	BUFLEVEL	0	RW	**BUFLEVEL Interrupt Enable**
	Enable/disable the BUFLEVEL interrupt			
19	BUFDATAV	0	RW	**BUFDATAV Interrupt Enable**
	Enable/disable the BUFDATAV interrupt			
18	DECERR	0	RW	**DECERR Interrupt Enable**
	Enable/disable the DECERR interrupt			
17	DEC	0	RW	**DEC Interrupt Enable**
	Enable/disable the DEC interrupt			
16	SCANCOMPLETE	0	RW	**SCANCOMPLETE Interrupt Enable**
	Enable/disable the SCANCOMPLETE interrupt			
15	CH15	0	RW	**CH15 Interrupt Enable**
	Enable/disable the CH15 interrupt			
14	CH14	0	RW	**CH14 Interrupt Enable**
	Enable/disable the CH14 interrupt			
13	CH13	0	RW	**CH13 Interrupt Enable**
	Enable/disable the CH13 interrupt			
12	CH12	0	RW	**CH12 Interrupt Enable**
	Enable/disable the CH12 interrupt			
11	CH11	0	RW	**CH11 Interrupt Enable**
	Enable/disable the CH11 interrupt			

Bit	Name	Reset	Access	Description
10	CH10	0	RW	**CH10 Interrupt Enable**
	Enable/disable the CH10 interrupt			
9	CH9	0	RW	**CH9 Interrupt Enable**
	Enable/disable the CH9 interrupt			
8	CH8	0	RW	**CH8 Interrupt Enable**
	Enable/disable the CH8 interrupt			
7	CH7	0	RW	**CH7 Interrupt Enable**
	Enable/disable the CH7 interrupt			
6	CH6	0	RW	**CH6 Interrupt Enable**
	Enable/disable the CH6 interrupt			
5	CH5	0	RW	**CH5 Interrupt Enable**
	Enable/disable the CH5 interrupt			
4	CH4	0	RW	**CH4 Interrupt Enable**
	Enable/disable the CH4 interrupt			
3	CH3	0	RW	**CH3 Interrupt Enable**
	Enable/disable the CH3 interrupt			
2	CH2	0	RW	**CH2 Interrupt Enable**
	Enable/disable the CH2 interrupt			
1	CH1	0	RW	**CH1 Interrupt Enable**
	Enable/disable the CH1 interrupt			
0	CH0	0	RW	**CH0 Interrupt Enable**
	Enable/disable the CH0 interrupt			

30.5.23 LESENSE_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																									0							
Access																									R							
Name																									CMD							

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	CMD	0	R	**CMD Register Busy**
	Set when the value written to CMD is being synchronized.			
6:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

30.5.24 LESENSE_ROUTEPEN - I/O Routing Register (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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									ALTEX7PEN	ALTEX6PEN	ALTEX5PEN	ALTEX4PEN	ALTEX3PEN	ALTEX2PEN	ALTEX1PEN	ALTEX0PEN	CH15PEN	CH14PEN	CH13PEN	CH12PEN	CH11PEN	CH10PEN	CH9PEN	CH8PEN	CH7PEN	CH6PEN	CH5PEN	CH4PEN	CH3PEN	CH2PEN	CH1PEN	CH0PEN

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	ALTEX7PEN	0	RW	**ALTEX7 Pin Enable**
	Set this bit to enable LESENSE ALTEX7 pin			
22	ALTEX6PEN	0	RW	**ALTEX6 Pin Enable**
	Set this bit to enable LESENSE ALTEX6 pin			
21	ALTEX5PEN	0	RW	**ALTEX5 Pin Enable**
	Set this bit to enable LESENSE ALTEX5 pin			
20	ALTEX4PEN	0	RW	**ALTEX4 Pin Enable**
	Set this bit to enable LESENSE ALTEX4 pin			
19	ALTEX3PEN	0	RW	**ALTEX3 Pin Enable**
	Set this bit to enable LESENSE ALTEX3 pin			
18	ALTEX2PEN	0	RW	**ALTEX2 Pin Enable**
	Set this bit to enable LESENSE ALTEX2 pin			
17	ALTEX1PEN	0	RW	**ALTEX1 Pin Enable**
	Set this bit to enable LESENSE ALTEX1 pin			
16	ALTEX0PEN	0	RW	**ALTEX0 Pin Enable**
	Set this bit to enable LESENSE ALTEX0 pin			
15	CH15PEN	0	RW	**CH15 Pin Enable**
	Set this bit to enable LESENSE CH15 pin			
14	CH14PEN	0	RW	**CH14 Pin Enable**
	Set this bit to enable LESENSE CH14 pin			
13	CH13PEN	0	RW	**CH13 Pin Enable**
	Set this bit to enable LESENSE CH13 pin			
12	CH12PEN	0	RW	**CH12 Pin Enable**
	Set this bit to enable LESENSE CH12 pin			
11	CH11PEN	0	RW	**CH11 Pin Enable**
	Set this bit to enable LESENSE CH11 pin			

Bit	Name	Reset	Access	Description
10	CH10PEN	0	RW	**CH10 Pin Enable**
	Set this bit to enable LESENSE CH10 pin			
9	CH9PEN	0	RW	**CH9 Pin Enable**
	Set this bit to enable LESENSE CH9 pin			
8	CH8PEN	0	RW	**CH8 Pin Enable**
	Set this bit to enable LESENSE CH8 pin			
7	CH7PEN	0	RW	**CH7 Pin Enable**
	Set this bit to enable LESENSE CH7 pin			
6	CH6PEN	0	RW	**CH6 Pin Enable**
	Set this bit to enable LESENSE CH6 pin			
5	CH5PEN	0	RW	**CH5 Pin Enable**
	Set this bit to enable LESENSE CH5 pin			
4	CH4PEN	0	RW	**CH4 Pin Enable**
	Set this bit to enable LESENSE CH4 pin			
3	CH3PEN	0	RW	**CH3 Pin Enable**
	Set this bit to enable LESENSE CH3 pin			
2	CH2PEN	0	RW	**CH2 Pin Enable**
	Set this bit to enable LESENSE CH2 pin			
1	CH1PEN	0	RW	**CH1 Pin Enable**
	Set this bit to enable LESENSE CH1 pin			
0	CH0PEN	0	RW	**CH0 Pin Enable**
	Set this bit to enable LESENSE CH0 pin			

30.5.25 LESENSE_STx_TCONFA - State Transition Configuration a (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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														0xX			X	X		0xXX					0xX				0xX			
Access														RW			RW	RW		RW					RW				RW			
Name														PRSACT			SETIF	CHAIN		NEXTSTATE					MASK				COMP			

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*		
18:16	PRSACT	0xX	RW	**Configure Transition Action**
	Configure which action to perform when sensor state equals COMP			

DECCTRL_PRSCNT = 0		
Mode	Value	Description
NONE	0	No PRS pulses generated
PRS0	1	Generate pulse on LESPRS0
PRS1	2	Generate pulse on LESPRS1
PRS01	3	Generate pulse on LESPRS0 and LESPRS1
PRS2	4	Generate pulse on LESPRS2
PRS02	5	Generate pulse on LESPRS0 and LESPRS2
PRS12	6	Generate pulse on LESPRS1 and LESPRS2
PRS012	7	Generate pulse on LESPRS0, LESPRS1 and LESPRS2
DECCTRL_PRSCNT = 1		
NONE	0	Do not count
UP	1	Count up
DOWN	2	Count down
PRS2	4	Generate pulse on LESPRS2
UPANDPRS2	5	Count up and generate pulse on LESPRS2.
DOWNANDPRS2	6	Count down and generate pulse on LESPRS2.

15	SETIF	X	RW	**Set Interrupt Flag Enable**
	Set interrupt flag when sensor state equals COMP			
14	CHAIN	X	RW	**Enable State Descriptor Chaining**
	When set, descriptor in the next location will also be evaluated			

Bit	Name	Reset	Access	Description
13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:8	NEXTSTATE	0xXX	RW	**Next State Index** Index of next state to be entered if the sensor state equals COMP
7:4	MASK	0xX	RW	**Sensor Mask** Set bit X to exclude sensor X from evaluation.
3:0	COMP	0xX	RW	**Sensor Compare Value** State transition is triggered when sensor state equals COMP

30.5.26 LESENSE_STx_TCONFB - State Transition Configuration B (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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														0xX			X			0xXX					0xX				0xX			
Access														RW			RW			RW					RW				RW			
Name														PRSACT			SETIF			NEXTSTATE					MASK				COMP			

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*		
18:16	PRSACT	0xX	RW	**Configure Transition Action**

Configure which action to perform when sensor state equals COMP

DECCTRL_PRSCNT = 0		
Mode	Value	Description
NONE	0	No PRS pulses generated
PRS0	1	Generate pulse on PRS0
PRS1	2	Generate pulse on PRS1
PRS01	3	Generate pulse on PRS0 and PRS1
PRS2	4	Generate pulse on PRS2
PRS02	5	Generate pulse on PRS0 and PRS2
PRS12	6	Generate pulse on PRS1 and PRS2
PRS012	7	Generate pulse on PRS0, PRS1 and PRS2
DECCTRL_PRSCNT = 1		
NONE	0	Do not count
UP	1	Count up
DOWN	2	Count down
PRS2	4	Generate pulse on PRS2
UPANDPRS2	5	Count up and generate pulse on PRS2.
DOWNANDPRS2	6	Count down and generate pulse on PRS2.

15	SETIF	X	RW	**Set Interrupt Flag**

Set interrupt flag when sensor state equals COMP

14:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
12:8	NEXTSTATE	0xXX	RW	**Next State Index**
	Index of next state to be entered if the sensor state equals COMP			
7:4	MASK	0xX	RW	**Sensor Mask**
	Set bit X to exclude sensor X from evaluation.			
3:0	COMP	0xX	RW	**Sensor Compare Value**
	State transition is triggered when sensor state equals COMP			

30.5.27 LESENSE_BUFx_DATA - Scan Results (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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													0xX				0xXXXX															
Access													R				RWH															
Name													DATASRC				DATA															

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	DATASRC	0xX	R	**Result Data Source**
	This bitfield contains the channel index for the sensor result in DATA.			
15:0	DATA	0xXXXX	RWH	**Scan Result Buffer**
	This bitfield contains the sensor result.			

30.5.28 LESENSE_CHx_TIMING - Scan Configuration (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

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									0xXXX										0xXX								0xXX					
Access									RW										RW								RW					
Name									MEASUREDLY										SAMPLEDLY								EXTIME					

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*		
23:14	MEASUREDLY	0xXXX	RW	**Set Measure Delay**
	Configure measure delay. Sensor measuring is delayed for MEASUREDLY EXCLK cycles.			
13:6	SAMPLEDLY	0xXX	RW	**Set Sample Delay**
	Configure sample delay. Sampling will occur after SAMPLEDLY SAMPLECLK cycles.			
5:0	EXTIME	0xXX	RW	**Set Excitation Time**
	Configure excitation time. Excitation will last EXTIME EXCLK cycles.			

30.5.29 LESENSE_CHx_INTERACT - Scan Configuration (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

Offset	Bit Position																															
0x244	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											X	X	X	0xX		0xX			0xX		0xXXX											
Access											RW	RW	RW	RW		RW			RW		RW											
Name											ALTEX	SAMPLECLK	EXCLK	EXMODE		SETIF			SAMPLE		THRES											

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	ALTEX	X	RW	**Use Alternative Excite Pin**
	If set, alternative excite pin will be used for excitation			
20	SAMPLECLK	X	RW	**Select Clock Used for Timing of Sample Delay**
	This bit is used to configure which clock is used for timing of SAMPLEDLY			

Value	Mode	Description
0	LFACLK	LFACLK will be used for timing
1	AUXHFRCO	AUXHFRCO will be used for timing

Bit	Name	Reset	Access	Description
19	EXCLK	X	RW	**Select Clock Used for Excitation Timing**
	This bit is used to configure which clock is used for timing of EXTIME and MEASUREDLY			

Value	Mode	Description
0	LFACLK	LFACLK will be used for timing
1	AUXHFRCO	AUXHFRCO will be used for timing

Bit	Name	Reset	Access	Description
18:17	EXMODE	0xX	RW	**Set GPIO Mode**
	GPIO mode for the excitation phase of the scan sequence. Note that DACOUT is only available on channels 0, 1, 2, 3, 12, 13, 14, 15			

Value	Mode	Description
0	DISABLE	Disabled
1	HIGH	Push Pull, GPIO is driven high
2	LOW	Push Pull, GPIO is driven low
3	DACOUT	VDAC output

Bit	Name	Reset	Access	Description
16:14	SETIF	0xX	RW	**Enable Interrupt Generation**
	Select interrupt generation mode for CHx interrupt flag.			

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	NONE		No interrupt is generated
	1	LEVEL		Set interrupt flag if the sensor triggers.
	2	POSEDGE		Set interrupt flag on positive edge of the sensor state
	3	NEGEDGE		Set interrupt flag on negative edge of the sensor state
	4	BOTHEDGES		Set interrupt flag on both edges of the sensor state
13:12	SAMPLE	0xX	RW	**Select Sample Mode**
	Select measurement to be used for evaluation			
	Value	Mode		Description
	0	ACMPCOUNT		Counter output will be used in evaluation
	1	ACMP		ACMP output will be used in evaluation
	2	ADC		ADC output will be used in evaluation
	3	ADCDIFF		Differential ADC output will be used in evaluation
11:0	THRES	0xXXX	RW	**ACMP Threshold or VDAC Data**
	Set threshold used for ACMP, or data used in VDAC conversion.			

30.5.30 LESENSE_CHx_EVAL - Scan Configuration (Async Reg)

For more information about asynchronous registers see 4.3 Access to Low Energy Peripherals (Asynchronous Registers).

Offset	Bit Position																															
0x248	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset										0xX		X	0xX		X	X	0xXXXX															
Access										RW		RW	RW		RW	RW	RWH															
Name										MODE		SCANRESINV	STRSAMPLE		DECODE	COMP	COMPTHRES															

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*		
22:21	MODE	0xX	RW	**Configure Evaluation Mode**

Select which evaluation mode to be used on the measurement result

Value	Mode	Description
0	THRES	Threshold comparison is used to evaluate sensor result
1	SLIDINGWIN	Sliding window is used to evaluate sensor result
2	STEPDET	Step detection is used to evaluate sensor result

Bit	Name	Reset	Access	Description
20	SCANRESINV	X	RW	**Enable Inversion of Result**

If set, the bit stored in SCANRES will be inverted.

Bit	Name	Reset	Access	Description
19:18	STRSAMPLE	0xX	RW	**Enable Storing of Sensor Sample in Result Buffer**

If set, the sensor sample value will be stored and available in the result buffer

Value	Mode	Description
0	DISABLE	Nothing will be stored in the result buffer.
1	DATA	The sensor sample data will be stored in the result buffer.
2	DATASRC	The data source (i.e., the channel) will be stored alongside the sensor sample data.

Bit	Name	Reset	Access	Description
17	DECODE	X	RW	**Send Result to Decoder**

If set, the result from this channel will be shifted into the decoder register.

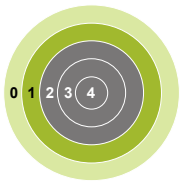
Bit	Name	Reset	Access	Description
16	COMP	X	RW	**Select Mode for Threshold Comparison**

Set compare mode for threshold comparisons (CHx_INTERACT_SAMPLE != ACMP and CHx_EVAL_MODE == THRES).

Value	Mode	Description
0	LESS	Comparison evaluates to 1 if sensor data is less than COMPTHRES.
1	GE	Comparison evaluates to 1 if sensor data is greater than or equal to COMPTHRES.

Bit	Name	Reset	Access	Description
15:0	COMPTHRES	0xXXXX	RWH	**Decision Threshold for Sensor Data**
	In threshold comparison mode, this bitfield is used to configure threshold used for comparison. In step detection mode, this bitfield is written by LESENSE, and contains the value from previous sensor measurement. In sliding window mode, this bitfield is written by LESENSE, and contains the window base for the given channel.			

31. 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.

31.1 Introduction

The GPCRC module is a slave peripheral that 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).

31.2 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

31.3 Functional Description

An overview of the GPCRC module is shown in Figure 31.1 GPCRC Overview on page 1167.

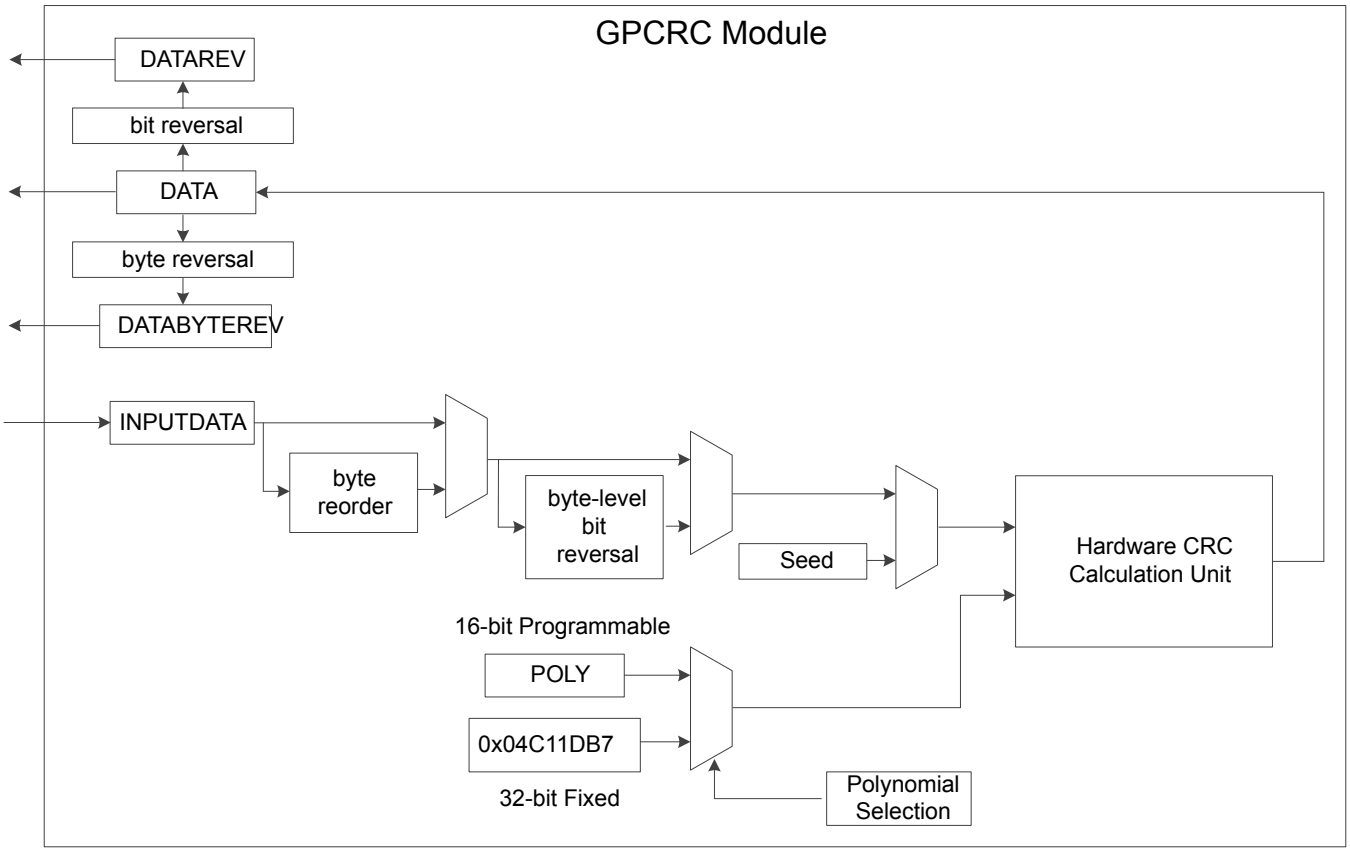


Figure 31.1. GPCRC Overview

31.3.1 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 31.2 Polynomial Representation on page 1168.

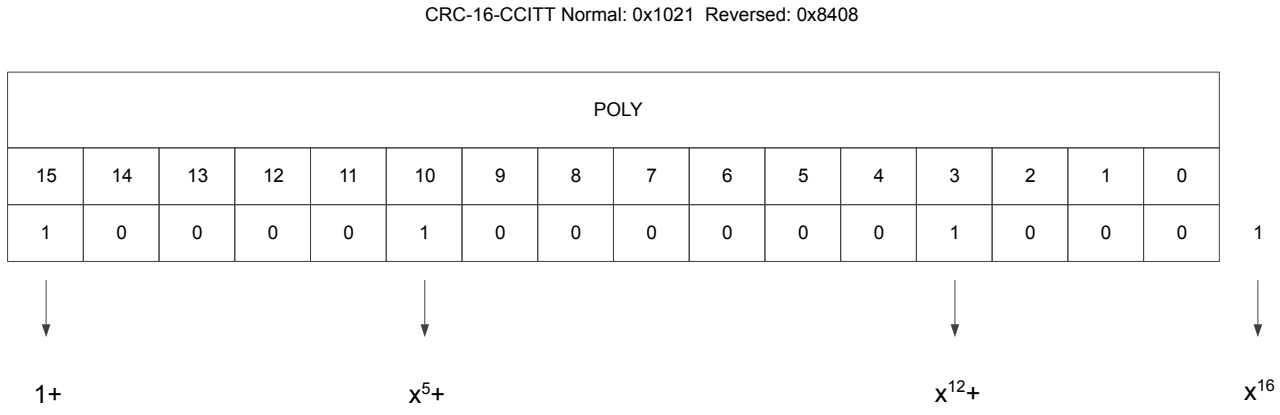


Figure 31.2. Polynomial Representation

31.3.2 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 needs one clock cycle, reading from GPCRC_DATA, GPCRC_DATAREV or GPCRC_DATABYTEREV register or writting to GPCRC_CMD register is halted while the calculation is in progress.

31.3.3 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.

31.3.4 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 avaliable words of data and use software writes to capture any remaining bytes.

31.3.5 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 31.3 Data Ordering Example - 32-bit MSB -first to LSB-first on page 1169. 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.

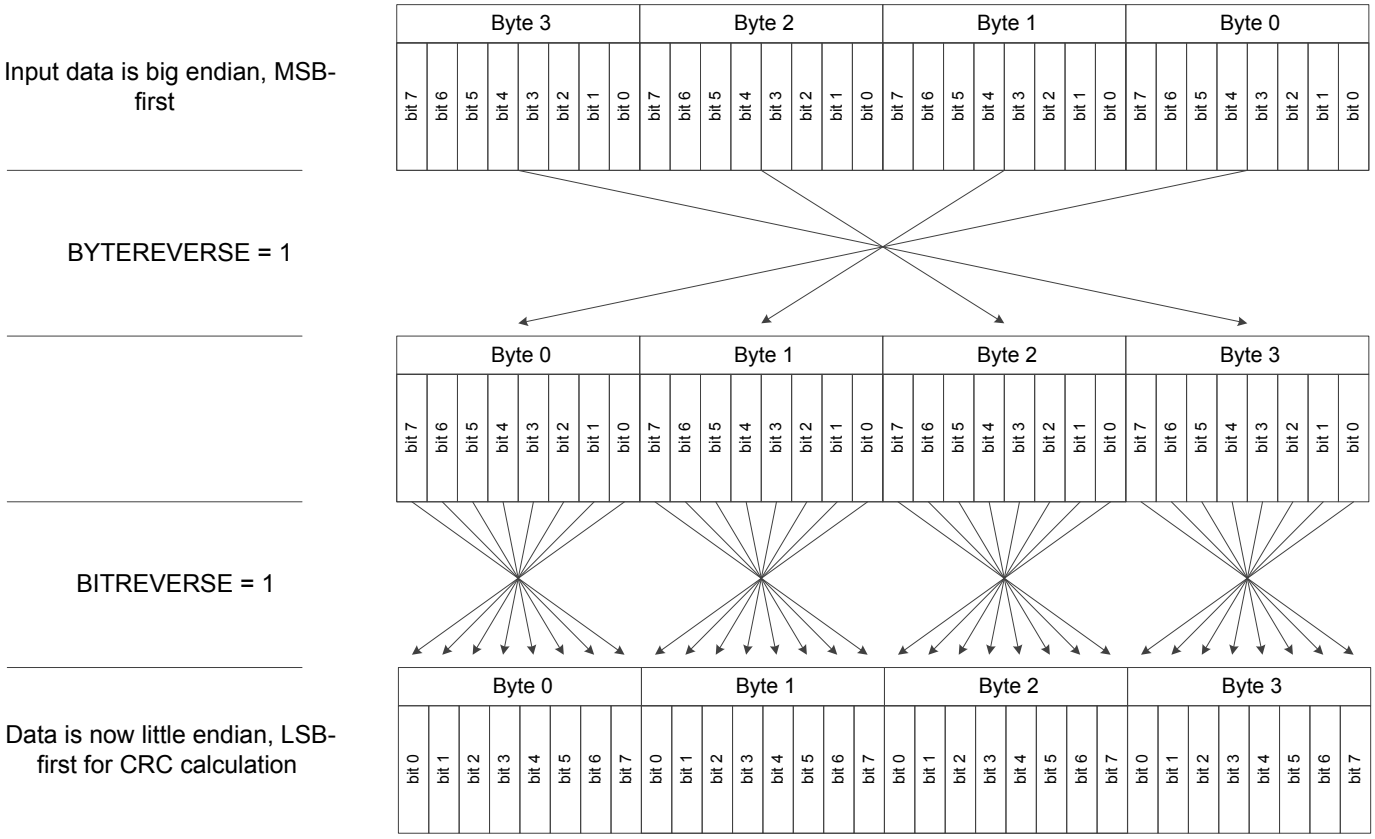


Figure 31.3. Data Ordering Example - 32-bit MSB -first to LSB-first

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 31.4 Data Ordering Example - 16-bit MSB -first to LSB-first on page 1170. 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.

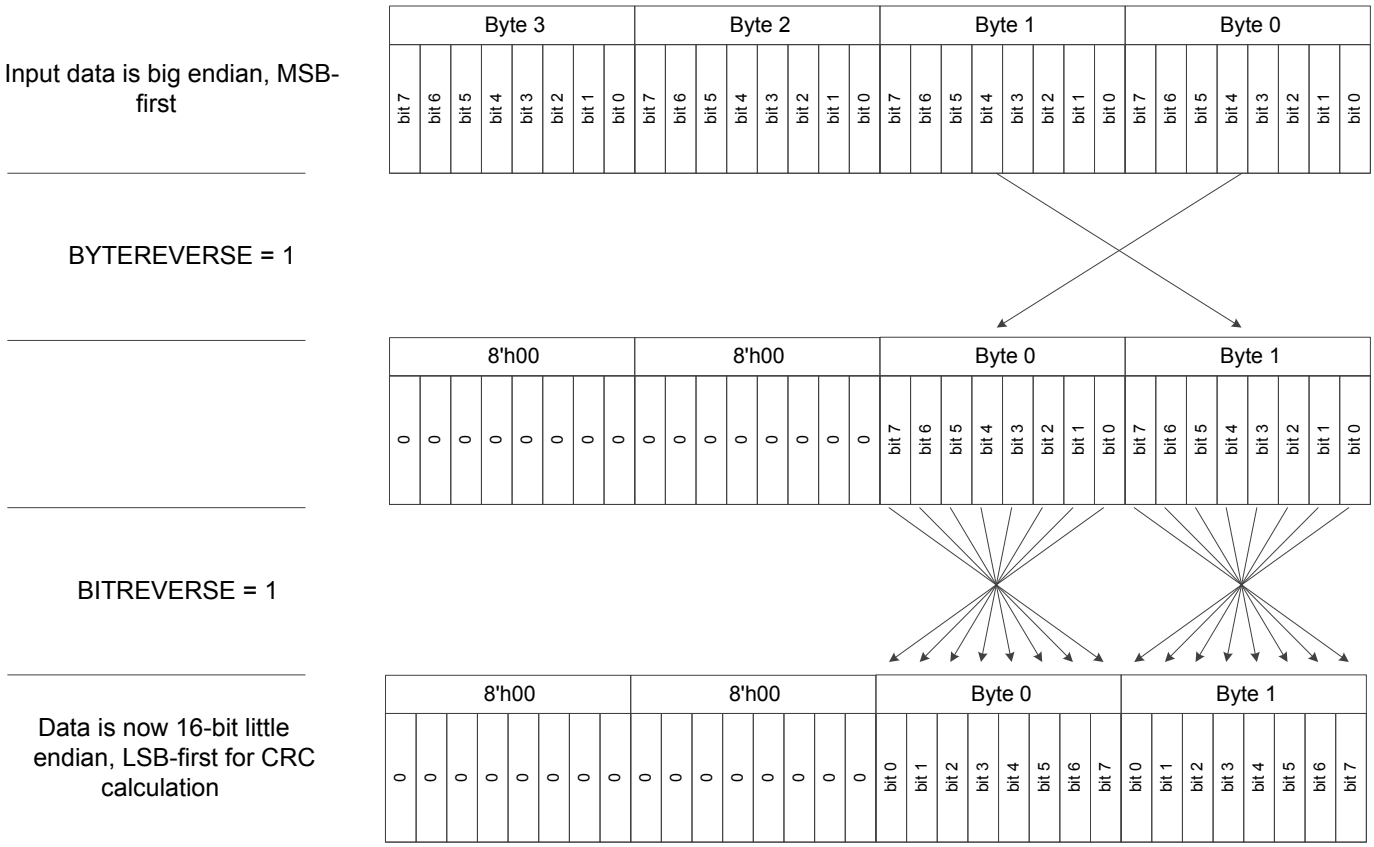


Figure 31.4. Data Ordering Example - 16-bit MSB -first to LSB-first

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 31.1 Byte-Level Bit Reversal and Byte Reordering Results (B3 B2 B1 B0 Input Order) on page 1170.

Table 31.1. 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

Note:

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.

31.4 Register Map

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

Offset	Name	Type	Description
0x000	GPCRC_CTRL	RW	Control Register
0x004	GPCRC_CMD	W1	Command Register
0x008	GPCRC_INIT	RWH	CRC Init Value
0x00C	GPCRC_POLY	RW	CRC Polynomial Value
0x010	GPCRC_INPUTDATA	W	Input 32-bit Data Register
0x014	GPCRC_INPUTDATAHWORD	W	Input 16-bit Data Register
0x018	GPCRC_INPUTDATABYTE	W	Input 8-bit Data Register
0x01C	GPCRC_DATA	R	CRC Data Register
0x020	GPCRC_DATAREV	R	CRC Data Reverse Register
0x024	GPCRC_DATABYTEREV	R	CRC Data Byte Reverse Register

31.5 Register Description

31.5.1 GPCRC_CTRL - 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																			0			0	0	0				0				0
Access																			RW			RW	RW	RW				RW				RW
Name																			AUTOINIT			BYTEREVERSE	BITREVERSE	BYTEMODE				POLYSEL				EN

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	0	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	0	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	0	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	0	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	0	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	16	16-bit CRC programmable polynomial selected

Bit	Name	Reset	Access	Description
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**CRC Functionality Enable**
	Enables CRC functionality.			

Value	Mode	Description
0	DISABLE	Disable CRC function. Reordering function is available, only BITREVERSE and BYTEREVERSE bits are configurable in this mode
1	ENABLE	Writes to inputdata registers result in CRC operations

31.5.2 GPCRC_CMD - Command 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																																0
Access																																W1
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	0	W1	**Initialization Enable**
	Writing 1 to this bit initialize the CRC by writing the INIT value in CRC_INIT to CRC_DATA.			

31.5.3 GPCRC_INIT - CRC Init Value

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	0x00000000																															
Access	RWH																															
Name	INIT																															

Bit	Name	Reset	Access	Description
31:0	INIT	0x00000000	RWH	**CRC Initialization Value**
	This value is loaded into CRC_DATA upon issuing the INIT command in CRC_CMD			

31.5.4 GPCRC_POLY - CRC Polynomial Value

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																	0x0000															
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	0x0000	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.			

31.5.5 GPCRC_INPUTDATA - Input 32-bit Data 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																0x00000000																
Access																W																
Name																INPUTDATA																

Bit	Name	Reset	Access	Description
31:0	INPUTDATA	0x00000000	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.			

31.5.6 GPCRC_INPUTDATAHWORD - Input 16-bit Data 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																	0x0000															
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	0x0000	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.			

31.5.7 GPCRC_INPUTDATABYTE - Input 8-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																									0x00							
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	0x00	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.			

31.5.8 GPCRC_DATA - CRC 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	0x00000000																															
Access	R																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	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.			

31.5.9 GPCRC_DATAREV - CRC Data Reverse 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	0x00000000																															
Access	R																															
Name	DATAREV																															

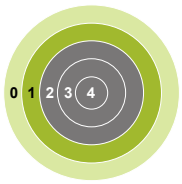
Bit	Name	Reset	Access	Description
31:0	DATAREV	0x00000000	R	**Data Reverse Value**
	Bit reversed version of CRC Data register. When a 32-bit CRC polynomial is selected, the reversal occurs on the entire 32-bit word. When a 16-bit CRC polynomial is selected, the bits [15:0] are reversed.			

31.5.10 GPCRC_DATABYTEREV - CRC Data Byte Reverse 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	0x00000000																															
Access	R																															
Name	DATABYTEREV																															

Bit	Name	Reset	Access	Description
31:0	DATABYTEREV	0x00000000	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}.			

32. TRNG - True Random Number Generator



Quick Facts

What?

The TRNG module is a non-deterministic random number generator based on a full hardware solution.

Why?

Secure cryptography commonly relies on randomly-generated numbers for key generation. Software solutions for random number generation do not usually produce results with enough entropy to satisfy existing standards. Dedicated hardware can provide suitable entropy in an energy-efficient, non-intrusive manner, while also relieving software burden.

How?

Ring oscillators and sampling logic combine to produce non-deterministic random numbers.

32.1 Introduction

The TRNG module is a non-deterministic random number generator based on a full hardware solution. The TRNG output passes the NIST 800-22 and AIS31 test suites.

32.2 Features

- Simple bus interface to access random numbers, control, and status registers
- 64 x 32-bit FIFO for random number access
- Interrupt sources from different FIFO, error, and noise alarm events
- Passes NIST 800-22 and AIS31
- Ready for NIST 800-90B
- Health tests compliant to NIST 800-90B and AIS31

32.3 Functional Description

Software drivers provided by Silicon Labs offer a simple API interface to the TRNG module. It is highly recommended to use the provided software librariesto access the TRNG module. An overview of the TRNG module is shown in Figure 32.1 TRNG Overview on page 1179.

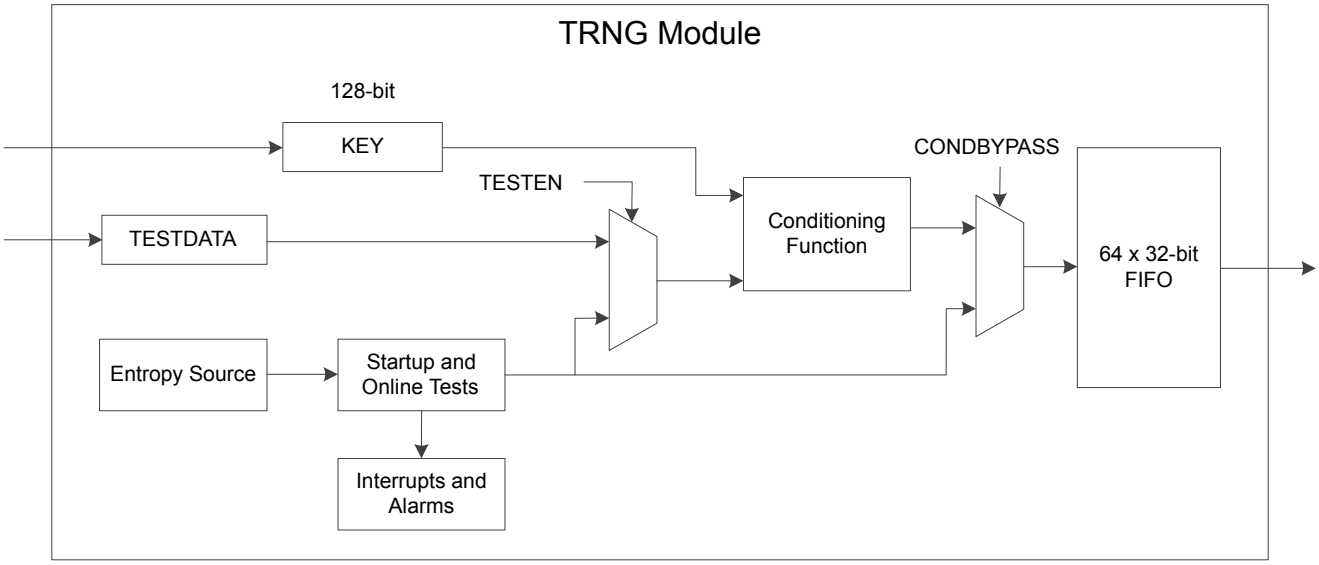


Figure 32.1. TRNG Overview

32.3.1 Built-In Tests

The TRNG module includes several built-in 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 (http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf) are implemented in hardware and run continuously on the data. All three tests have corresponding interrupt flags that can optionally be used to generate a system interrupt.

The AIS31 Online Test described in section 5.5.3 of https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf is also implemented in hardware, and runs continuously on the data. Both the preliminary noise alarm and the noise alarm are optionally available as interrupt sources from the TRNG module. If a noise alarm occurs, the TRNG will be shut down, and must be reset with a software reset.

Additionally, the NIST-800-90B and AIS31 startup tests may be optionally enabled or disabled by software. The NIST-800-90B startup test is enabled if the CONTROL_BYPNIST bit is cleared to 0. The AIS31 startup test is likewise enabled when CONTROL_BYPAIS31 is cleared to 0. If either the NIST-800-90B or AIS31 startup tests are enabled, no data will be written to the output FIFO until the startup requirements for these tests pass.

32.3.2 FIFO Interface

The TRNG module includes a 64-word output FIFO to hold the output data as it becomes available. The number of 32-bit words available in the FIFO may be checked at any time by reading the FIFOLEVEL register. When the FIFO is completely filled, the TRNG will be shut down, the STATUS_FULLIF flag will be set, and no further data will be written to the FIFO until the FIFO has been flushed. Data may be read from the FIFO one word (32-bits) at a time via the FIFO register. The STATUS_FULLIF flag is cleared upon reading the FIFOLEVEL register.

32.3.3 Data Format - Byte Ordering

All cryptographic data is handled following the big-endian format (AES standard). The first byte (lowest address) of the data is the Most Significant Byte (MSB). However, the bus interfaces on the core use little endian format for internal byte ordering within a 32-bit word. The Least Significant Byte (LSB) within a word is stored at the lowest address.

For example, a 128-bit block 0x00112233445566778899AABBCCDDEEFF is read from the FIFO in the following order:

Word 1 = 0x33221100

Word 2 = 0x77665544

Word 3 = 0xBBAA9988

Word 4 = 0xFFDDEECC

This is important to note when checking the conditioning function for validity. The KEY registers also follow this standard, with KEY0 holding the MSB of a 128-bit value and KEY3 holding the LSB.

32.3.4 TRNG Usage

It is highly recommended to use the software libraries provided by Silicon Labs to access the TRNG module. The information in the following sections are reference for users who choose to write their own low-level software drivers.

32.3.4.1 Checking the Conditioning Function

The conditioning function receives 512 bits from the entropy source and generates 128 bits of output. The conditioning function can be tested by writing a known key and known data into the block with test mode enabled, then validating against the expected output. The sequence of operations to test the conditioning function is as follows:

1. Apply a software reset by setting CONTROL_SOFTRESET to 1, then clearing it to 0.
2. Configure the CONTROL register. Important configuration options include:
   - Enable test mode by setting CONTROL_TESTEN to 1.
   - Ensure the conditioning function is used by clearning CONTROL_CONDBYPASS to 0.
3. Write the key into registers KEY0, KEY1, KEY2, and KEY3.
4. Write the 512 bits of known data to TESTDATA 32 bits at a time, polling for STATUS_TESTDATABUSY = 0 after each write.
5. Read the 128-bit result from the FIFO 32 bits at a time.

Table 32.1 Known-Answer Test for Conditioning Function on page 1181 shows an example with a given key, known data input, and expected output (taken from section F.2.1 in http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf).

Table 32.1. Known-Answer Test for Conditioning Function

	128-bit Format	32-bit Bus Format
Key	0x2B7E151628AED2A6ABF7158809CF4F3C	0x16157E2B 0xA6D2AE28 0x8815F7AB 0x3C4FCF09
Input	0x6BC0BCE12A459991E134741A7F9E1925 0xAE2D8A571E03AC9C9EB76FAC45AF8E51 0x30C81C46A35CE411E5FBC1191A0A52EF 0xF69F2445DF4F9B17AD2B417BE66C3710	0xE1BCC06B 0x9199452A 0x1A7434E1 0x25199E7F 0x578A2DAE 0x9CAC031E 0xAC6FB79E 0x518EAF45 0x461CC830 0x11E45CA3 0x19C1FBE5 0xEF520A1A 0x45249FF6 0x179B4FDF 0x7B412BAD 0x10376CE6
Expected output	0x3FF1CAA1681FAC09120ECA307586E1A7	0xA1CAF13F 0x09AC1F68 0x30CA0E12 0xA7E18675

32.3.4.2 Checking the Entropy Source

The entropy source may be checked using the three built in test sources: repetition count, 64-sample adaptive proportion, and 4096-sample adaptive proportion. The entropy source may be tested using the following sequence:

1. Apply a software reset by setting CONTROL_SOFTRESET to 1, then clearing it to 0.
2. Configure the CONTROL register. Important configuration options include:
   - Disable test mode by clearing CONTROL_TESTEN to 0.
   - Bypass the conditioning function by setting CONTROL_CONDBYPASS to 1.
   - Enable the TRNG by setting CONTROL_ENABLE to 1.
3. Check the FIFOLEVEL register to monitor the amount of generated random numbers or wait for the STATUS_FULLIF flag to set, indicating the FIFO is full. Note that STATUS_FULLIF may be configured to generate an interrupt if desired.
4. When FIFOLEVEL has reached the expected value or when STATUS_FULLIF is set, read the random numbers using the FIFO register. Those values can be discarded.
5. Continue reading and discarding the random data until at least 4097 x 2 bits (257 x 32-bit words) have been read. This ensures that enough time has passed for the longest test to run.
6. Check the STATUS register for error flags.

32.3.4.3 Programming a Random Key

1. Check the FIFOLEVEL register to monitor the amount of generated random numbers or wait for the STATUS_FULLIF flag to set, indicating the FIFO is full. Note that STATUS_FULLIF may be configured to generate an interrupt if desired.
2. When FIFOLEVEL has reached the expected value or when STATUS_FULLIF is set, read the random numbers using the FIFO register.
3. Use four 32-bit random values to program a random key to the KEY0, KEY1, KEY2, and KEY3 registers.
4. Apply a software reset by setting CONTROL_SOFTRESET to 1, then clearing it to 0. This will flush the FIFO data.

32.3.4.4 Reading Samples

1. Check the FIFOLEVEL register to monitor the amount of generated random numbers or wait for the STATUS_FULLIF flag to set, indicating the FIFO is full. Note that STATUS_FULLIF may be configured to generate an interrupt if desired.
2. When FIFOLEVEL has reached the expected value or when STATUS_FULLIF is set, read the random numbers using the FIFO register.

32.4 Register Map

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

Offset	Name	Type	Description
0x000	TRNGn_CONTROL	RW	Main Control Register
0x004	TRNGn_FIFOLEVEL	R(a)	FIFO Level Register
0x00C	TRNGn_FIFODEPTH	R	FIFO Depth Register
0x010	TRNGn_KEY0	RW	Key Register 0
0x014	TRNGn_KEY1	RW	Key Register 1
0x018	TRNGn_KEY2	RW	Key Register 2
0x01C	TRNGn_KEY3	RW	Key Register 3
0x020	TRNGn_TESTDATA	RW	Test Data Register
0x030	TRNGn_STATUS	RWH	Status Register
0x034	TRNGn_INITWAITVAL	RW	Initial Wait Counter
0x100	TRNGn_FIFO	R(a)	FIFO Data
0x300	TRNGn_CORECLKCONTROL	RW	Core Clock Control Register

32.5 Register Description

32.5.1 TRNGn_CONTROL - Main 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																			0	0	0	0	0	0	0	0	0	0	0	0		0
Access																			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW
Name																			BYPAIS31	BYPNIST	FORCERUN	ALMIEN	PREIEN	SOFTRESET	FULLIEN	APT4096IEN	APT64IEN	REPCOUNTIEN	CONDBYPASS	TESTEN		ENABLE

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	BYPAIS31	0	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.

Bit	Name	Reset	Access	Description
12	BYPNIST	0	RW	**NIST Start-up Test Bypass.**

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.

Bit	Name	Reset	Access	Description
11	FORCERUN	0	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

Bit	Name	Reset	Access	Description
10	ALMIEN	0	RW	**Interrupt enable for AIS31 noise alarm**

Enable/disable AIS31 noise alarm interrupt.

Bit	Name	Reset	Access	Description
9	PREIEN	0	RW	**Interrupt enable for AIS31 preliminary noise alarm**

Enable/disable AIS31 preliminary noise alarm interrupt.

Bit	Name	Reset	Access	Description
8	SOFTRESET	0	RW	**Software Reset**

Set to reset the module. This bit is not cleared automatically.

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	NORMAL		Module not in reset
	1	RESET		The continuous test, the conditioning function and the FIFO are reset
7	FULLIEN	0	RW	**Interrupt Enable for FIFO Full**
	Enable/Disable FIFO full interrupt.			
6	APT4096IEN	0	RW	**Interrupt Enable for Adaptive Proportion Test Failure (4096-sample Window)**
	Enable/Disable 4096-sample Adaptive Proportion test failure interrupt.			
5	APT64IEN	0	RW	**Interrupt Enable for Adaptive Proportion Test Failure (64-sample Window)**
	Enable/Disable 64-sample Adaptive Proportion test failure interrupt.			
4	REPCOUNTIEN	0	RW	**Interrupt Enable for Repetition Count Test Failure**
	Enable/Disable Repetition Count Test failure interrupt.			
3	CONDBYPASS	0	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	0	RW	**Test Enable**
	Selects the input for the conditioning function and continuous tests.			
	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	0	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

32.5.2 TRNGn_FIFOLEVEL - FIFO Level Register (Actionable Reads)

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	0x00000000																															
Access	R																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	R	**FIFO Level**
	Number of 32-bit words of random data available in the FIFO. The STATUS_FULLIF flag is cleared when FIFOLEVEL is read.			

32.5.3 TRNGn_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	0x00000040																															
Access	R																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000040	R	**FIFO Depth.**
	Maximum number of 32-bit words that can be stored in the FIFO.			

32.5.4 TRNGn_KEY0 - Key Register 0

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	0x00000000																															
Access	RW																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	RW	**Key 0**
	AES Key 32-bit sub-word 0 (MSB).			

32.5.5 TRNGn_KEY1 - Key Register 1

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	0x00000000																															
Access	RW																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	RW	**Key 1**
	AES Key 32-bit sub-word 1.			

32.5.6 TRNGn_KEY2 - Key Register 2

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	0x00000000																															
Access	RW																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	RW	**Key 2**
	AES Key 32-bit sub-word 2.			

32.5.7 TRNGn_KEY3 - Key Register 3

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	0x00000000																															
Access	RW																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	RW	**Key 3**
	AES Key 32-bit sub-word 3 (LSB).			

32.5.8 TRNGn_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	0x00000000																															
Access	RW																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	RW	**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.			

32.5.9 TRNGn_STATUS - 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																							0	0	0	0	0	0				0
Access																							R	RWH	R	R	R	R				R
Name																							ALMIF	PREIF	FULLIF	APT4096IF	APT64IF	REPCOUNTIF				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	0	R	**AIS31 Noise Alarm interrupt status**
	Set when a noise alarm is detected in the AIS31 online test.			
8	PREIF	0	RWH	**AIS31 Preliminary Noise Alarm interrupt status**
	Set when a preliminary noise alarm is detected in the AIS31 online test.			
7	FULLIF	0	R	**FIFO Full Interrupt Status**
	Set when the FIFO is full. The STATUS_FULLIF flag is cleared by reading FIFOLEVEL.			
6	APT4096IF	0	R	**Adaptive Proportion test failure (4096-sample window) interrupt status**
	Set when an Adaptive Proportion test (4096-sample window) failure occurs.			
5	APT64IF	0	R	**Adaptive Proportion test failure (64-sample window) interrupt status**
	Set when an Adaptive Proportion test (64-sample window) failure occurs.			
4	REPCOUNTIF	0	R	**Repetition Count Test Interrupt Status**
	Set when a Repetition Count Test failure occurs.			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	TESTDATABUSY	0	R	**Test Data Busy**
	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.

32.5.10 TRNGn_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																							0x3FF								
Access																							RW									
Name																							VALUE									

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	VALUE	0x3FF	RW	**Wait counter value**
	Number of clock cycles to wait before sampling data from the noise source.			

32.5.11 TRNGn_FIFO - FIFO Data (Actionable Reads)

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																0x00000000																
Access																R																
Name																VALUE																

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	R	**FIFO Read Data**
	Data may be read from the FIFO 32 bits at a time using this register.			

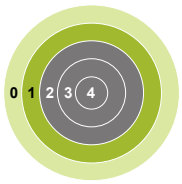
32.5.12 TRNGn_CORECLKCONTROL - Core Clock Control Register

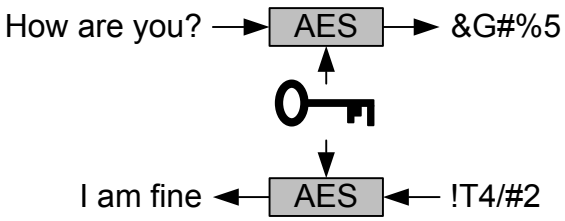
Offset	Bit Position																														
0x300	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x0				0
Access																											RW					RW
Name																											CORECLKPRESC					CORECLKDIS

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	CORECLKPRESC	0x0	RW	**Clock division factor of CORECLKPRESC+1**
	Clock prescaler value for TRNG core clock			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	CORECLKDIS	0	RW	**Core Clock Disable**
	Set this bit to disable clock to TRNG core. Clock to TRNG APB registers remains running			

Value	Mode	Description
0	ENABLED	CLK to TRNG core is enabled
1	DISABLED	CLK to TRNG core is disabled

33. CRYPTO - Crypto Accelerator





Quick Facts

What?

A fast and energy efficient autonomous hardware accelerator for AES encryption and decryption with 128- or 256-bit keys, ECC over prime and binary Galois finite fields, SHA-1, SHA-224 and SHA-256.

Why?

Efficient cryptography with little or no CPU intervention helps to meet the speed and energy demands of the application. Hardware implementations are generally more secure against side-channel attacks than software implementations.

How?

Programmable sequences of instructions on big numbers allow fast processing with little CPU intervention.

33.1 Introduction

The CRYPTO module allows efficient acceleration of common cryptographic operations and allows these to be used efficiently with a low CPU load. Operations performed by CRYPTO can be set up as a sequence of instructions on a set of 128-bit, 256-bit or 512-bit registers to implement or accelerate Elliptic Curve Cryptography (ECC), SHA-1, SHA-224, SHA-256, and various block cipher modes based on the Advanced Encryption Standard, also known as AES (FIPS-197).

CRYPTO is capable of autonomously fetching data, performing cipher operations and storing data across multiple blocks. When the source data is not a multiple of 16 bytes (128 bits), Zero-padding can be included in the last block. Block operations such as Counter Mode (CTR), Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB) are easily implemented. Block Cipher modes of operation such as Electronic Code Book (ECB), Counter Mode (CTR), Cipher Block Chaining (CBC), CBC-MAC (CBC Message Authentication Code), CCM (Counter with CBC-MAC) and GCM (Galois Counter mode) are easily implemented.

CRYPTO is delivered with an extensive software library in Simplicity Studio that implements all major cryptographic algorithms, including but not limited to AES, SHA-1, SHA-2, ECC, and legacy algorithms DES, 3DES, MD4, MD5 and RC4. The implementation accelerates the algorithms using CRYPTO when possible.

33.2 Features

- Efficient AES core
  - Encryption/decryption using 128-bit key (54 clock cycles) or 256-bit key (75 clock cycles)
  - Key buffer
  - Supports autonomous cipher block modes (e.g. ECB, CTR, CBC, PCBC, CFB, CBC-MAC, GMAC, CCM, CCM* and GCM) across multiple blocks
- Accelerated SHA-1, SHA-224 and SHA-256
- Accelerated Elliptic Curve Cryptography (ECC)
  - Binary and Prime fields
  - Supports NIST recommended curves: P-192, P-224, P-256, K-163, K-233, B-163, and B-233
- Galois/Counter Mode (GCM)
  - ALU operations on GCM GF(2^128) field
- Flexible 256-bit ALU and sequencer
  - 5 general purpose 256-bit registers
  - Supports ADD, SUB, MUL, shift, XOR, etc.
  - Up to 20 instructions can be chained to implement various block cipher modes
- Efficient operation
  - DMA request signals for data read and write
  - Optional XOR Data write
  - Interrupt on finished operations
- Extensive software support
  - Extensive software library in Simplicity Studio
  - Implements all major cryptographic algorithms: AES, SHA-1, SHA-2, and ECC
  - Implements legacy algorithms: DES, 3DES, MD4, MD5, and RC4
  - Hardware accelerated when possible

33.3 Usage and Programming Interface

Many security systems fail due to mistakes in the implementation. Therefore implementations should be left to experts in cryptographic algorithms.

To solve this, the module is supported by an hardened cryptography software library and API delivered through Silicon Labs' Simplicity Studio. The software API is a frontend for performing all supported cryptographic operations, and must be used to receive prompt support.

33.4 Functional Description

A block diagram of the CRYPTO module is shown in Figure 33.1 CRYPTO Overview on page 1194.

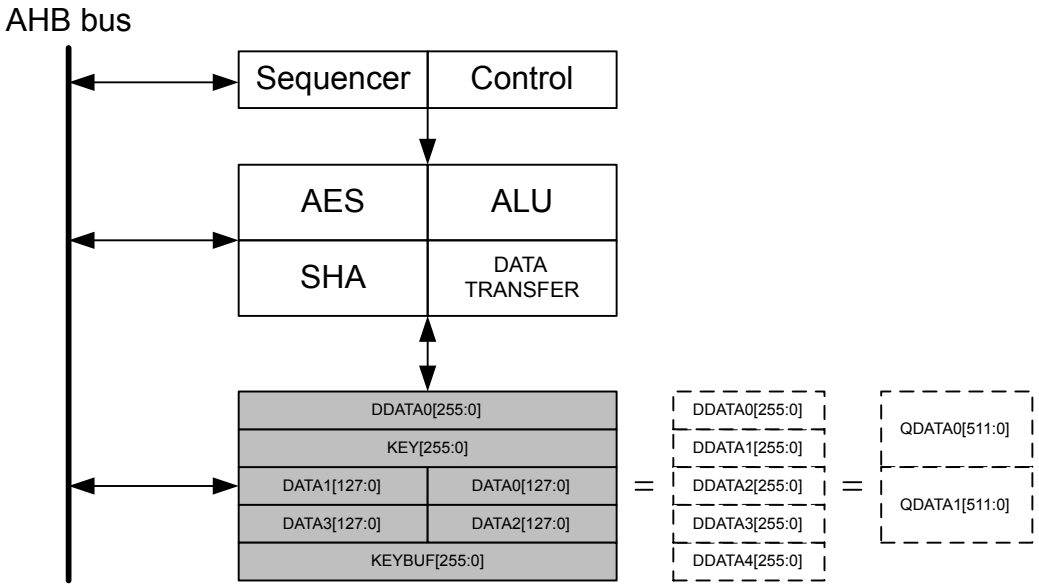


Figure 33.1. CRYPTO Overview

33.4.1 Data and Key Registers

The CRYPTO module contains five 256-bit registers. Accelerators are implemented through instructions operating on these registers, either by copying data between registers and external components through DMA, or by executing instructions on the registers.

Depending on the instruction, the registers can be accessed as 128-bit, 256-bit or 512-bit registers. The registers can also be accessed through different interface registers to achieve different results.

When writing to and reading from the CRYPTO_DATAx, CRYPTO_KEY, CRYPTO_KEYBUF, CRYPTO_DDATAx and CRYPTO_QDATAx registers, the least significant part is accessed first and the most significant part last, see Figure 33.2 CRYPTO Data and Key Register Operation on page 1196. The same is the case for the XOR and byte-access registers for DATA0 and DATA1. It is important to note that some of the 256-bit registers are composed of the 128-bit registers, and both the 512-bit registers are composed of the 256-bit registers.

Note: From here on, the 128, 256 and 512-bit registers are named DATAx, DDATAx, QDATAx, etc, And the access-points to these registers are named CRYPTO_DATAx, CRYPTO_DDATAx, CRYPTO_QDATAx, etc.

DATA0 can be accessed through CRYPTO_DATA0 (32-bit), CRYPTO_DATA0XOR (32-bit), CRYPTO_DATA0BYTE (8-bit) and CRYPTO_DATA0XORBYTE (8-bit). Direct access to bytes 12 - 15 is available through CRYPTO_DATA0BYTE12-15 (8-bit). The DATA0XOR (in CRYPTO_DATA0XOR) is used for XOR'ing a value with the current value in DATA0. This is used in a large variety of block cipher modes. All of these registers operate on DATA0.

DATA1 can be accessed through CRYPTO_DATA1 (32-bit) and CRYPTO_DATA1BYTE (8-bit).

The remaining data registers have regular 32-bit access through their respective registers. Note that all data registers require a full read or write to be fully accessed. This means that the 128-bit registers need four 32-bit reads/writes, the 256-bit registers need 8 reads/writes and the 512-bit registers need 16 reads/writes. For a read, if all read accesses are not done, the register will end up as a shifted version of the original value.

Note: For byte-wise data accesses (DDATAxBYTE, DATAxBYTE, etc.), all reads and writes must be performed in groups of 4, due to internal buffering and shifting of 32 bits at a time. Accessing a number of bytes that is not a multiple of four can cause data incoherency in all of the data registers.

The KEY and KEYBUF registers are 256 bit wide when AES256 is set in CRYPTO_CTRL. Else they are 128 bit wide. When used as a part of DDATAx and QDATAx, they are always 256 bit wide.

The registers DDATA0BIG and QDATA1BIG produce byte-swapped versions of DDATA0 and QDATA1 respectively. These may be used when a computation requires byte-swapping. An example of this is SHA computation, where data needs to be changed to big endian before CRYPTO can work with it. Little endian data is then loaded in through QDATA1BIG and the resulting little endian hash can be read out from DDATA0BIG, see 33.4.5 SHA.

Except for KEYBUF, the contents of all data registers are lost when going to EM2.

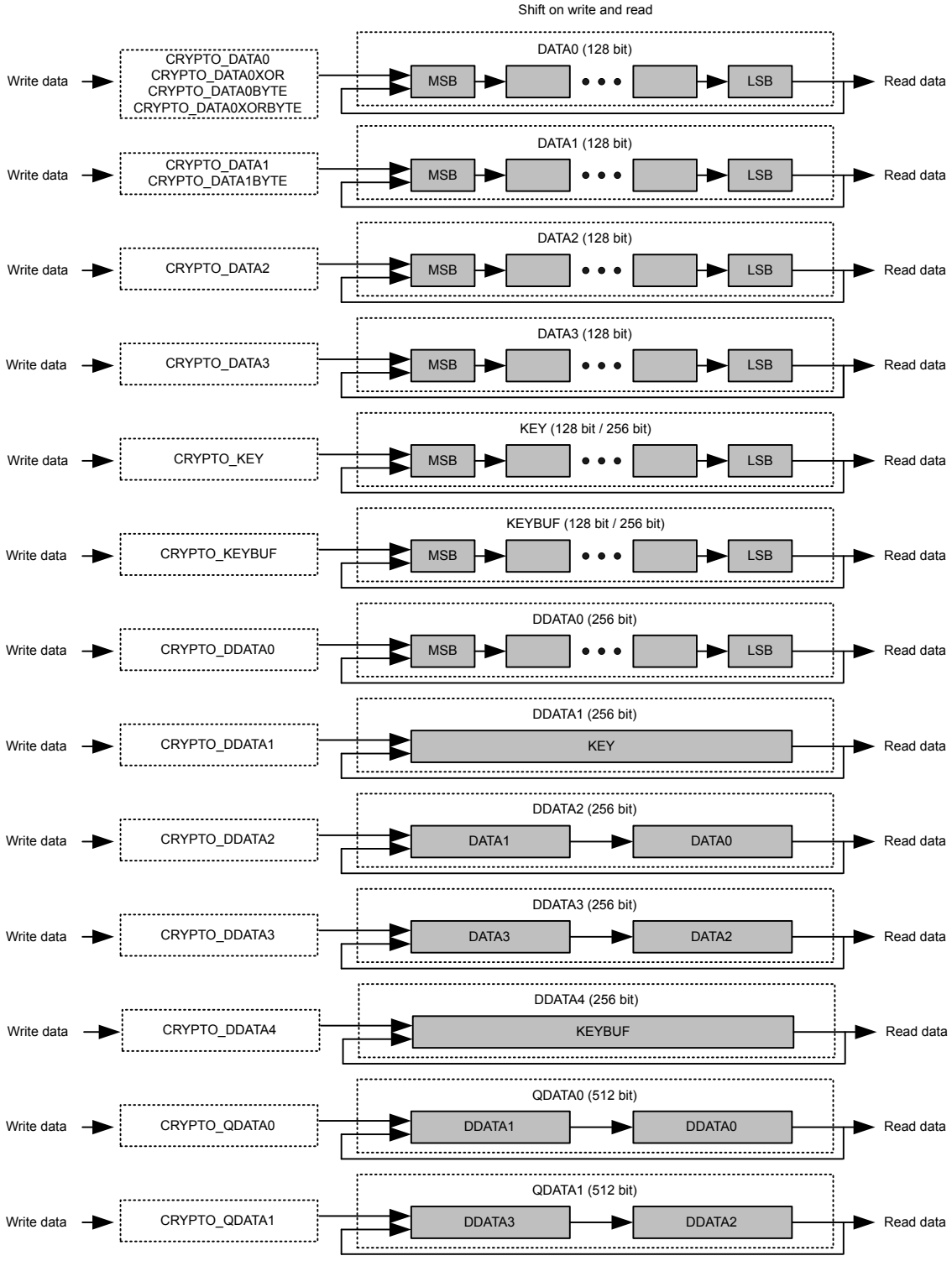


Figure 33.2. CRYPTO Data and Key Register Operation

33.4.1.1 DATA0 Zero

DATA0ZERO in CRYPTO_DSTATUS contains status flags indicating if any 32-bit blocks within DATA0 is 0. For example, if DATA0[95:64] is equal to 0x00000000, ZERO64TO95 is set.

33.4.1.2 DDATA0 and DDATA1 Quick Observation

DDATA0LSBS in CRYPTO_DSTATUS shows the 4 least significant bits in DDATA0. DDATA0MSBS in CRYPTO_DSTATUS shows the 4 most significant bits of DDATA0, while DDATA1MSB in CRYPTO_DSTATUS shows the msb of DDATA1. These observation bitfields are useful for determining the sign of the value in the data registers without having to read out the full register data register values

The 4 bits observed by DDATA0MSBS will change depending on RESULTWIDTH in CRYPTO_WAC. When using 260-bit results, DDATA0MSBS shows bits 259-256, when using 256-bit results, it is bits 255-252, and for 128-bit results, bits 127-124 can be observed. When RESULTWIDTH is 260 bits, the 4 most significant bits, e.g. bits 259-256 are also available in CRYPTO_DDATA0BYTE32, where they can also be written. Using this register is the only way of inputting the upper 4 bits of a 260-bit number to CRYPTO.

33.4.1.3 Result Width

RESULTWIDTH in CRYPTO_WAC determines the width of the operation when performing arithmetic/shift instructions with CRYPTO. Using less wide results will reduce the current consumption of the CRYPTO module. The higher-order bits that are beyond the selected result width are ignored in the computation of arithmetic/shift operations, however, these higher-order bits will be undefined in the result of such instructions.

When RESULTWIDTH=260BIT, all DDATA registers effectively become 260 bits wide, so that the upper 4 bits are not lost when transferring data from DDATA0 to the other DDATA registers. Likewise, the arithmetic/shift instructions shall consider the full 260-bit values of DDATA0-DDATA4 when used as operation inputs. Note that DDATA0 is the only 260-bit register of which MSBs can be observed/written. The upper 4 bits are observed through DDATA0MSBS in CRYPTO_DSTATUS or through CRYPTO_DDATA0BYTE32. For all DDATAx registers, the extra MSBs are cleared when DDATAx is written. Furthermore, for a particular x, a write to DDATAx or any of its aliased registers will cause DDATAx MSBs to be cleared. Note, writing to KEY/KEYBUF will only clear MSBs of DDATA1/DDATA4 when AES256 mode is set. Likewise, writing to DATA0/DATA2 will not clear DDATA2/DDATA3 MSBs.

Since the DATA0-DATA3 registers are always 128-bit, all bit positions greater than 128 are interpreted as 0 when RESULTWIDTH is greater than 128 bits. However, the assignment instructions DATAxTODDATAy will not zero-out the upper 128 bits of the DDATAy target. Instead, those upper words become undefined after such operations.

33.4.2 Instructions and Execution

The CRYPTO module implements a set of instructions in order to load and manipulate data effectively. These instructions are grouped into four types:

- ALU instructions - arithmetic and logical bitwise operations
- Transfer instructions - moving data between registers and external peripherals like DMA
- Conditional instructions - conditionally execute instructions based on context
- Special instructions - various crypto and support instructions

A single instruction can be executed by writing INSTR in CRYPTO_CMD. This will execute the instruction, and the interface of CRYPTO will be locked until the execution has completed. Multiple commands can safely be issued after each other by the CPU as long as NOBUSYSTALL in CRYPTO_CTRL is not set. If CRYPTO gets a new command or a data access request while busy it will then stall the bus, and execute the new command as soon as it is done with the previous one. Note, there are some exceptions to this rule. For example, see 33.4.8 DMA.

Stalling of the bus can be disabled by setting NOBUSYSTALL in CRYPTO_CTRL, however manipulating (reading or writing) registers while running an instruction will result in undefined behaviour. Additionally, if NOBUSYSTALL=0 and a new command or data access request is made while the CRYPTO is simultaneously performing a data transfer instruction, it is possible for system lockup due to bus stalling loops. The safest approach is to always check if an instruction is running by looking at INSTRRUNNING in CRYPTO_STATUS.

Note that this automatic stalling feature does not apply to automated CRYPTO instruction sequences (described next), since there may be cycle delays between individual instructions for which bus accesses are not prevented. For sequences, always check the SEQRUNNING status bit or the SEQDONE interrupt flag to ensure the sequence is finished before attempting CRYPTO register accesses.

33.4.2.1 Sequences

For executing a set of instructions, it is more efficient to load them into the CRYPTO module and run them as a sequence. This is done by writing the instructions into CRYPTO_SEQ0-CRYPTO_SEQ4, and marking the end of the instruction sequence with either an END or an EXEC instruction. The END simply means end-of-instructions, while writing EXEC means end-of-instructions and execute immediately.

The five registers allow up to 20 instructions to be loaded. To start execution, either end the instructions with an EXEC instruction, or set SEQSTART in CRYPTO_CMD. CRYPTO will then execute the instructions, starting in CRYPTO_SEQ0, and ending at the first END instruction. SEQRUNNING in CRYPTO_STATUS is set while the sequence is running, and the interrupt flag SEQDONE in CRYPTO_IF will be set when the sequence has completed.

A sequence can be stopped by issuing the SEQSTOP command in the CRYPTO_CMD register. This command also clears the state of ongoing CRYPTO instructions including DMA access. Check SEQRUNNING in CRYPTO_STATUS after issuing the SEQSTOP command flag to make sure any ongoing sequence/transfer has completed before accessing data registers again.

33.4.2.2 Available Instructions

The available ALU instructions are listed in Table 33.1 ALU Instructions on page 1199, long instructions are listed in Table 33.2 Long Instructions on page 1200, data transfer instructions are listed in Table 33.3 Transfer Instructions on page 1200, conditional instructions are listed in Table 33.4 Conditional Instructions on page 1201 and special instructions are listed in Table 33.5 Special Instructions on page 1201. The tables explains the side-effects of the instructions and shows which registers are affected. V0 and V1 in the instructions descriptions can be any of the DDATAx registers and a selection of the DATAx registers. They can be selected using the SELDDATAxDDATAy, SELDATAxDDATAy, SELDDATAxDATAy and SELDATAxDATAy instructions. The first register in the instruction will be selected for V0, and the second for V1. This configuration stays even when the sequence is complete, and can also be set up front. The currently selected V0 and V1 can be read V0 and V1 in CRYPTO_CSTATUS.

Table 33.1. ALU Instructions

Instruction	Description	Constraints/Notes
ADD	DDATA0 = V0 + V1	If V0 != DDATA0, then V1 != DDATA0
ADDO	DDATA0 = V0 + V1	Carry is only set, not cleared. If V0 != DDATA0, then V1 != DDATA0
ADDC	DDATA0 = V0 + V1 + carry	If V0 != DDATA0, then V1 != DDATA0
ADDIC	DDATA0 = V0 + V1 + carry << 128	If V0 != DDATA0, then V1 != DDATA0. If resultwidth is 128b, then carry is undefined
MADD	DDATA0 = (V0 + V1) mod P	If V0 != DDATA0, then V1 != DDATA0
MADD32	DDATA0[i] = V0[i] + V1[i]. Word-wise addition	carry is not modified. If V0 != DDATA0, then V1 != DDATA0
SUB	DDATA0 = V0 - V1	V1 != DDATA0. If V1 is 128b and resultwidth > 128b, then upper 128b are unknown
SUBC	DDATA0 = V0 - V1 - carry	V1 != DDATA0. If V1 is 128b and resultwidth > 128b, then upper 128b are unknown
MSUB	DDATA0 = (V0 - V1) mod P	V1 != DDATA0. If V1 is 128b and resultwidth > 128b, then upper 128b are unknown
MUL	DDATA0 = DDATA1 * V1. See 33.4.2.3 MULx Details	V1 != DDATA0,DDATA1
MULC	DDATA0 = DDATA1 * V1 + (DDATA0 << MULWIDTH). See 33.4.2.3 MULx Details	V1 != DDATA0,DDATA1
MMUL	DDATA0 = (DDATA1 * V1) mod P	V1 != DDATA0,DDATA1
MULO	DDATA0 = DDATA1 * V1. See 33.4.2.3 MULx Details	V1 != DDATA0,DDATA1. Carry is only set, not cleared
SHL	DDATA0 = V0 << 1	If V0 is 128b and resultwidth is 260b, then upper 4b are unknown
SHLC	DDATA0 = V0 << 1 | carry	If V0 is 128b and resultwidth is 260b, then upper 4b are unknown
SHLB	DDATA0 = V0 << 1 | V0[resultwidth-1]	If V0 is 128b and resultwidth is 260b, then upper 4b are unknown
SHL1	DDATA0 = V0 << 1 | 1	If V0 is 128b and resultwidth is 260b, then upper 4b are unknown
SHR	DDATA0 = V0 >> 1	
SHRC	DDATA0 = V0 >> 1 | carry << resultwidth-1	
SHRB	DDATA0 = V0 >> 1 | V0[0] << resultwidth-1	

Instruction	Description	Constraints/Notes
SHR1	DDATA0 = V0 >> 1 | 1 << resultwidth-1	
SHRA	DDATA0 = V0 >> 1 | V0[resultwidth-1] << result-width-1	
CLR	DDATA0 = 0	
XOR	DDATA0 = V0 ^ V1	If V0 != DDATA0, then V1 != DDATA0
INV	DDATA0 = ~V0	
CSET	CARRY = 1	
CCLR	CARRY = 0	
BBSWAP128	DDATA0[127:0] = bbswap(V0[127:0])	See 33.4.2.6 BBSWAP128 Instruction
INC	DDATA0 = DDATA0 + 1	
DEC	DDATA0 = DDATA0 - 1	

Table 33.2. Long Instructions

Instruc-tion	Operation	Constraints/Notes
LADD	{DDATA1,DDATA0} = V0 + V1, if V0 != DDATA0. {DDATA1,DDATA0} = {DDATA1,DDATA0} + V1, if V0 = DDATA0	If V0 != DDATA0, then V1 != DDATA0
LADDO	{DDATA1,DDATA0} = V0 + V1, if V0 != DDATA0. {DDATA1,DDATA0} = {DDATA1,DDATA0} + V1, if V0 = DDATA0	Carry is only set, not cleared. If V0 != DDATA0, then V1 != DDATA0
LADDC	{DDATA1,DDATA0} = V0 + V1 + carry, if V0 != DDATA0. {DDATA1,DDATA0} = {DDATA1,DDATA0} + V1 + carry, if V0 = DDATA0	If V0 != DDATA0, then V1 != DDATA0
LADDIC	{DDATA1,DDATA0} = V0 + V1 + carry << 256, if V0 != DDATA0. {DDATA1,DDATA0} = {DDATA1,DDATA0} + V1 + carry << 256, if V0 = DDATA0	If V0 != DDATA0, then V1 != DDATA0
LSUB	{DDATA1,DDATA0} = V0 - V1, if V0 != DDATA0. {DDATA1,DDATA0} = {DDATA1,DDATA0} - V1, if V0 = DDATA0	V1 != DDATA0. If V1 is 128b, then upper 128b are unknown
LSUBC	{DDATA1,DDATA0} = V0 - V1 - carry, if V0 != DDATA0. {DDATA1,DDATA0} = {DDATA1,DDATA0} - V1 - carry, if V0 = DDATA0	V1 != DDATA0. If V1 is 128b, then upper 128b are unknown
LMUL	{DDATA1,DDATA0} = DDATA1 * V1	V1 != DDATA0,DDATA1
LMULO	{DDATA1,DDATA0} = DDATA1 * V1	V1 != DDATA0,DDATA1. Carry is only set, not cleared
LINC	{DDATA1,DDATA0} = {DDATA1,DDATA0} + 1	
LDEC	{DDATA1,DDATA0} = {DDATA1,DDATA0} - 1	

Table 33.3. Transfer Instructions

Instruction	Operation	Constraints/Notes
DATATODMA0	DMA = DATAX, DMA request DMA0RD	DATAX = DATA0, DDATA0, DDATA0BIG, QDATA0 as defined by DMA0RSEL
DMA0TODATA	DATAX = DMA, DMA request DATA0WR	DATAX = DATA0, DDATA0, DDATA0BIG, QDATA0
DMA0TODATAXOR	DATA0 = DATA0 ^ DMA, DMA request DA-TA0XORWR	

Instruction	Operation	Constraints/Notes
DATATODMA1	DMA = DATAX, DMA request DMA1RD	DATAX = DATA1, DDATA1, QDATA1, QDATA1BIG as defined by DMA1RSEL
DMA1TODATA	DATAX = DMA, DMA request DATA0WR	DATAX = DATA1, DDATA1, QDATA1, QDATA1BIG
DATAxTODATAy	DATAy = DATAx	
DATAxTODATA0XOR	DATA0 = DATA0 ^ DATAx	If resultwidth is 128b, then carry is undefined
DATAxTODATA0XOR-LEN	DATA0 = DATA0 ^ (DATAx & (2**LENGTH-1))	LENGTH is LENGTHA or LENGTHB depending on active part of sequence. If resultwidth is 128b, then carry is undefined
DDATAxTODDATAy	DDATAy = DDATAx	
DDATAxHTODATA1	DATA1 = DDATAx[255:128]	Bits DDATA2[259:256] become undefined
DDATAxLTODATAy	DATAy = DDATAx[127:0]	
SELDDATAxDDATAy	Use DDATAx as V0, DDATAy as V1	x = 0,1,2,3,4; y = 0,1,2,3,4
SELDATAxDDATAy	Use DATAx as V0, DDATAy as V1	x = 0,1,2; y = 0,1,2,3,4
SELDDATAxDATAy	Use DDATAx as V0, DATAy as V1	x = 0,1,2,3,4; y = 0,1
SELDATAxDATAy	Use DATAx as V0, DATAy as V1	x = 0,1,2; y = 0,1

Table 33.4. Conditional Instructions

Instruction	Operation	Constraints
EXECIFA	Execute following instructions if in part A of sequence	
EXECIFB	Execute following instructions if in part B of sequence	
EXECIFNLAST	Execute following instructions if not in last iteration of sequence	
EXECIFLAST	Execute following instructions if in last iteration of sequence	
EXECIFCARRY	Execute following instructions if carry bit is set	
EXECIFNCARRY	Execute following instructions if carry bit not is set	
EXECALWAYS	Always execute following instructions	

Table 33.5. Special Instructions

Instruction	Operation
END	Ends execution.
EXEC	When written to CRYPTO_SEQx register, automatically triggers execution of all instruction up to this point.
AESENC	DATA0 = AESENC(DATA0)
AESDEC	DATA0 = AESDEC(DATA0)
SHA	DDATA0 = SHA(Q1)
DATA1INC	DATA1 = inc(DATA1). See 33.4.2.5 DATA1INC and DATA1INCCLR Instructions
DATA1INCCLR	DATA1 = clearinc(DATA1). See 33.4.2.5 DATA1INC and DATA1INCCLR Instructions

33.4.2.3 MULx Details

For the MULx instructions (not MMUL), MULWIDTH in CRYPTO_WAC specifies the width of operands DDATA1 (and sometimes V1). This is useful in order to optimize performance because multiplications take the same number of cycles as the bits in the operands plus a couple of cycles for setup.

As with the other ALU instructions, RESULTWIDTH limits the width of the final result of the MULx and MMUL instructions.

33.4.2.4 Long Instruction Details

For the Long instructions listed in Table 33.2 Long Instructions on page 1200, RESULTWIDTH in CRYPTO_WAC is ignored and is treated as if it were set to 512 bits. Likewise, MULWIDTH is also ignored and is treated as if it were set to 256 bits.

33.4.2.5 DATA1INC and DATA1INCCLR Instructions

DATA1INC and DATA1INCCLR operate on the 1, 2, 3 or 4 most significant bytes in DATA1, depending on INCWIDTH in CRYPTO_CTRL. DATA1INC increments these bytes in big endian, while DATA1INCCLR clears the bytes.

33.4.2.6 BBSWAP128 Instruction

The BBSWAP128 instruction copies the contents of the V0 operand to DDATA0 while swapping the bits of the lower 16 bytes. The operand is not changed. This operation is required for GCM. See 33.4.7 GCM and GMAC

33.4.2.7 Carry

The carry output from most instructions can be observed through the CARRY bit in CRYPTO_DSTATUS. Shift-instructions set CARRY to the value that is shifted out of the register, addition and multiplication set it on register overflow, and subtraction sets it on borrow, e.g. underflow.

In addition to generating carry information, some instructions also use the current value of CARRY. ADDC, SUBC, SHLC and SHRC all use carry to generate the result. For all of these instructions, carry allows a program to chain instructions together to operate on bigger numbers than allowed by CRYPTO. For example, by chaining first an ADD, and then an ADDC which uses the carry from the ADD operation, two 512-bit numbers can be added. By chaining more instructions, even larger numbers can be manipulated.

Other uses of CARRY include observation. To check if a register is 0, one can subtract 1 using the DEC instruction, and check if goes negative by checking the CARRY bit. CARRY can be set manually and in CRYPTO programs using the CSET and CCLR instructions, which set and clear the CARRY bit.

The MULC instruction does not use CARRY like the other carry instructions (i.e., instructions ending in 'C' such as 'ADDC'), but rather preserves the old contents of the multiplication register.

33.4.3 Repeated Sequence

To maximize efficiency, it is desirable to be able to run a set of instructions over multiple blocks of data autonomously. To repeat a sequence over a larger set of data, set LENGTHA in CRYPTO_SEQCTRL to the number of bytes in the set, and BLOCKSIZE to the size of the blocks in the set. The sequence will then be repeated N times, where N is LENGTHA / BLOCKSIZE if LENGTHA is a multiple of BLOCKSIZE, or ceiling(LENGTHA / BLOCKSIZE) if not. In the latter case, data written by DMA will be zero-padded up to BLOCKSIZE if it is written to a register which has a size equal to BLOCKSIZE. One notable exception is when LENGTHA is 0. In this case the sequence will still execute once, but the block transfer instructions will not execute.

Note: If DMAxRSEL in CRYPTO_CTRL selects a register that is smaller than the specified blocksize, DATATODMAx/DMAxTODATA instructions will not use the full blocksize, but will only transfer enough data to empty/fill the register once. For example, if BLOCKSIZE is set to 64B and DMA0RSEL=DDATA0, the instruction DATATODMA0 will only read 32B instead of 64B. The processing of LENGTHA/B will continue as if all 64B had been transferred.

A repeated sequence can also be made do slightly different operations on different parts of the data set. A sequence can be divided into two parts; part A, and part B. By configuring LENGTHA in CRYPTO_SEQCTRL to the length of part A, and LENGTHB in CRYPTO_SEQCTRLB to the length of part B, CRYPTO will first run iterations over part A, knowing it is A, and then part B, knowing it is part B. By using the conditional instructions listed in Table 33.4 Conditional Instructions on page 1201, a program can execute different instructions depending on whether it is in part A or part B.

33.4.4 AES

The AES core operates on data in the 128-bit register DATA0 using the either a 128-bit or 256-bit key from the KEY register. The key width is specified by AES256 in CRYPTO_CTRL. AES operations are implemented as the AESENC and AESDEC instructions, for AES encryption and AES decryption respectively. An overview of the AES functionality is shown in Figure 33.3 CRYPTO AES Overview on page 1203.

AES encryption and decryption enables various block cipher modes like ECB, CTR, CBC, PCBC, CFB, OFB, CBC-MAC, GMAC, CCM, CCM*, and GCM.

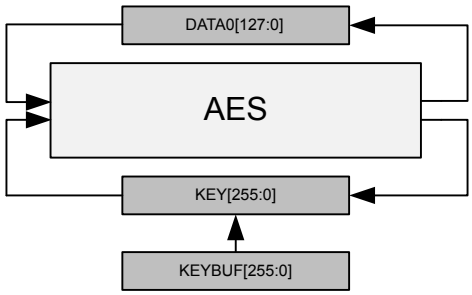


Figure 33.3. CRYPTO AES Overview

The input data before encryption is called the PlainText and output from the encryption is called CipherText. For encryption, the key is called PlainKey. After encryption, the resulting key in the KEY registers is the CipherKey. This key must be loaded into the KEY registers prior to the decryption. After one decryption, the resulting key will be the PlainKey. The resulting PlainKey/CipherKey is only dependent on the value in the KEY registers before encryption/decryption. The resulting keys and data are shown in Figure 33.4 CRYPTO Key and Data Definitions on page 1203.

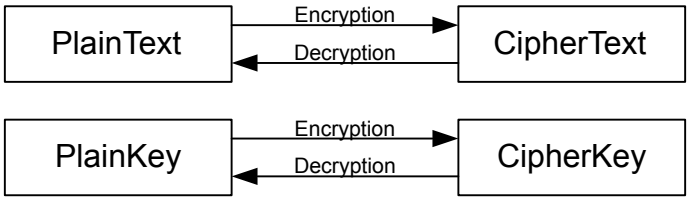


Figure 33.4. CRYPTO Key and Data Definitions

The KEY is by default loaded from KEYBUF prior to each AESENC or AESDEC instruction. If the KEY is not to be overwritten, key buffering should be disabled (KEYBUFDIS in CRYPTO_CTRL). Disabling key buffering also allows the use of key loading through DMA.

The data and key orientation in the CRYPTO registers are shown in Figure 33.5 CRYPTO Data and Key Orientation as Defined in the Advanced Encryption Standard on page 1204.

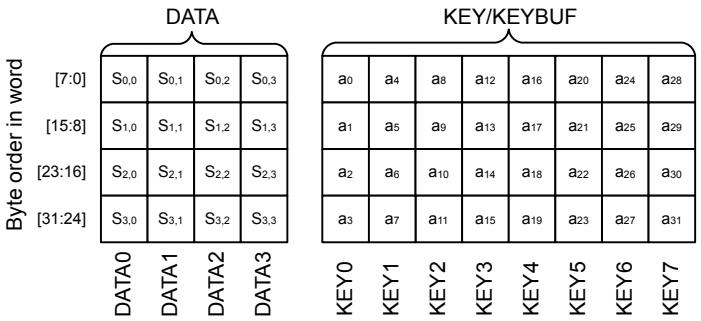


Figure 33.5. CRYPTO Data and Key Orientation as Defined in the Advanced Encryption Standard

33.4.5 SHA

The CRYPTO SHA instruction implements SHA-1 with a 160-bit digest or SHA-2 with a 224-bit digest (SHA-224) or 256-bit digest (SHA-256). Depending on SHAMODE in CRYPTO_CTRL, SHA-1, SHA-224 or SHA-256 will be run on the data in QDATA1, and the result will be put on DDATA0. The contents in QDATA1 will be destroyed in the process.

To run SHA on a dataset, it must first be pre-processed by appending a bit '1' to the message, then padding the data with '0' bits until the message length in bits modulo 512 is 448. Then append the length of the message before pre-processing as a 64-bit big-endian integer. This pre-processing is known as MD-strengthening, and must be done by software before processing with the CRYPTO module.

The pre-processed data can now be run through the CRYPTO module. Begin by writing the values listed in Table 33.6 SHA Init Values on page 1205 to CRYPTO_DDATA1 from top to bottom, then execute the instructions listed in Table 33.7 SHA Preparations on page 1205.

Table 33.6. SHA Init Values

SHA-1	SHA-224	SHA-256
0x67452301	0xC1059ED8	0x6A09E667
0xEFCDAB89	0x367CD507	0xBB67AE85
0x98BADCFE	0x3070DD17	0x3C6EF372
0x10325476	0xF70E5939	0xA54FF53A
0xC3D2E1F0	0xFFC00B31	0x510E527F
0x00000000	0x68581511	0x9B05688C
0x00000000	0x64F98FA7	0x1F83D9AB
0x00000000	0xBEFA4FA4	0x5BE0CD19

Table 33.7. SHA Preparations

STEP	ACTION	Description
STEP0	DDATA1TODDATA0	Copy init data to DDATA0
STEP1	SELDDATA0DDATA1	Select DDATA0 and DDATA1 as operands for SHA instruction

Then, for each 512-bit block, write the block to CRYPTO_QDATA1BIG, execute the instructions listed in Table 33.8 SHA for 512-bit Block on page 1205.

Table 33.8. SHA for 512-bit Block

STEP	ACTION	Description
STEP0	SHA	Perform SHA operation on data in QDATA1
STEP1	MADD32	Accumulate with previous data in DDATA1
STEP2	DDATA0TODDATA1	Copy hash to DDATA1

After the last iteration, the resulting hash can be read out from CRYPTO_DDATA0BIG.

33.4.6 ECC

The CRYPTO module implements support for Elliptic Curve Cryptography through the modular instructions MADD, MMUL and MSUB, which perform modular addition, multiplication and subtraction respectively. The instructions can operate on a set of both prime fields GF(p) and binary fields GF(2^m).

The type of modular arithmetic used and the modulus for the modular operations are specified by MODOP and MODULUS in CRYPTO_WAC respectively. Changing these in the middle of an operation leads to undefined behaviour.

33.4.7 GCM and GMAC

CRYPTO implements support for Galois/Counter Mode (GCM), and also Galois Message Authentication Code (GMAC), by providing AES instructions and allowing multiplication on the field GF(2^128) defined by the polynomial x^128 + x^7 + x^2 + x + 1.

Note: BBSWAP128 needs to be applied to both operands and the result of the MMUL instruction when using it for GCM and GMAC

Efficient sequencer programs can be set up to perform GCM authentication and encryption/decryption on data from either DMA, or CPU. To achieve a single-pass solution, LENGTHA in CRYPTO_SEQCTRL is set to the length of the authentication part, and LENGTHB is set to the length of the rest of the message. Conditional instructions can then be used to make sure the two parts of the message are processed correctly. A similar approach is used to implement CCM.

33.4.8 DMA

The CRYPTO module has 5 DMA request signals (see Table 33.9 DMA Signals on page 1206) split over 2 internal DMA channels: DMA0 and DMA1. These DMA channels are not associated with channel 0 and 1 of the system DMA, and any system DMA channel can serve any of the 5 DMA requests. See the DMA chapter for information on how to configure the system DMA.

The DMA signals are set through the use of DMA oriented instructions, and cleared by reading or writing the respective CRYPTO data registers.

Table 33.9. DMA Signals

Name	Set on	Cleared on
DMA0WR	Instruction DMA0TODATA, and DMA0TODATAXOR if COMBDMA0WEREQ in CRYPTO_CTRL is set	Full CRYPTO_DATA0, CRYPTO_DDATA0, CRYPTO_DDATA0BIG or CRYPTO_QDATA0 write, or CRYPTO_DDATA0XOR if COMBDMA0WEDMAREQ in CRYPTO_CTRL is set
DMA0XORWR	Instruction DMA0TODATAXOR	Full CRYPTO_DATA0XOR write
DMA0RD	Instructions DATATODMA0	Full CRYPTO_DATA0, CRYPTO_DDATA0, CRYPTO_DDATA0BIG or CRYPTO_QDATA0 read, depending on DMA0MODE in CRYPTO_CTRL
DMA1WR	Instructions DMA1TODATA	Full CRYPTO_DATA1, CRYPTO_DDATA1, CRYPTO_QDATA1 or CRYPTO_QDATA1BIG write
DMA1RD	Instructions DATATODMA1	Full CRYPTO_DATA1, CRYPTO_DDATA1, CRYPTO_QDATA1 or CRYPTO_QDATA1BIG read, depending on DMA1MODE in CRYPTO_CTRL

Note: DMAxRSEL in CRYPTO_CTRL has to be set to the data registers that are to be read using the respective DMA channels on a DATATODMAx instruction. As an important note, DMAxRSEL in CRYPTO_CTRL selects what is read from **any** of the selectable read registers during an ongoing DATATODMAx transfer .

When a DMA oriented CRYPTO instruction is used (either through a STEP in a Sequence or through CRYPTO_CMD), the corresponding DMA signal is set. The instruction is complete when the entire source/destination is read/written (e.g. if DMA0TODATA is used, the operation is complete when a total of 128 valid bits have been written through the CRYPTO_DATA0 register). DMAACTIVE in CRYPTO_STATUS is set while CRYPTO is working on a DMA-related instruction, e.g. waiting for the DMA to read or write data to CRYPTO (see 33.4.8.1 DMA Initial Bytes Skip).

Normally, when a sequence or instruction is executed, access to most CRYPTO registers will stall the CPU or DMA that is trying to access CRYPTO until the operation is done, preventing accesses to CRYPTO that could potentially interfere with an operation. During DMA operations, all non-DMA registers are writeable and readable, but progress through the DMA operation will only be tracked with the registers targeted by the DMA operation (i.e., if the DMA operation is supposed to transfer 3 words to DATA0, the DMA can first choose to transfer data to e.g. DATA3, and then fulfill the transfer to DATA0).

Because the bus interface to CRYPTO is normally locked outside of DMA transfers, a wrongly set up DMA transfer (e.g., transferring one byte too many) may lock up the interface.. One way to assist in debugging such issues can be setting NOBUSYSTALL in CRYPTO_CTRL. This will prevent any stall on CRYPTO register accesses during sequences and instructions. Use this option with care, as modifying a register that is being used by CRYPTO can lead to undefined behavior.

33.4.8.1 DMA Initial Bytes Skip

The DMA must be configured to use 32-bit transfer size. This normally would imply that the source data must be aligned to a 4 byte address boundary. However, it is possible to skip the initial bytes (1 to 3) when using DMA to write to DATA0 or DATA1 through a CRYPTO instruction operation. The number of bytes to skip are set in DMA0SKIP and DMA1SKIP in CRYPTO_SEQCTRL. This implies that if DMA0SKIP is set to another value than 0, the initial DMA access will require 5 DMA transfers, even though only 4x32-bit is required.

Note: Any valid unused bytes from a previous DMA write will be used before new DMA data is requested. This data is invalidated by using STOP in CRYPTO_CMD.

33.4.8.2 DMA Unaligned Read/Write

Except for DATA0 and DATA1, which can be loaded bytewise using the CRYPTO_DATA0BYTE, CRYPTO_DATA0XORBYTE and CRYPTO_DATA1BYTE registers, the CRYPTO data registers are loaded 32-bits at a time. Special care must be taken when using the DMA and the data buffer is not aligned to a 32-bit address, because the DMA does not directly support 32-bit unaligned accesses.

As an example, let an in-memory 16-byte data buffer start at address 4*N + M and end at the byte before. 4*N + 16 + M, where M is between 0 and 3 inclusive. With an M=0, we have fully aligned accesses, and everything is fine. For M>0 however, the access is unaligned. If M=1, that means that the first 32-bit aligned word of the memory buffer contains 1 byte before the buffer, and 3 bytes of the buffer. Similarly, the last 32-bit aligned word of the memory buffer contains the last byte of the buffer, and three bytes after the buffer.

When doing an unaligned read, we want to only pass the 16 bytes of the buffer to the CRYPTO module. Not the N bytes before in the 32-bit aligned word, and not the 4-N words at the end. To achieve this, set DxDMAREADMODE in CRYPTO_CTRL to either UNALIGNEDFULL or UNALIGNEDLENLIMIT, and set DATAxDMASKIP in CRYPTO_SEQCTRL equal to N. When reading in data using a DMA-oriented instruction to DATAx, DDATAx or QDATAx, the read will now only contain the 16 bytes, and not the N bytes before or 4-N words after. Note that in this case, the DMA has to be set up to transfer 5 32-bit words instead of the effective 4.

Being able to read unaligned data does not solve all cases however. If data is to be written back to the buffer after passing through CRYPTO, e.g. when doing an in-place encryption or decryption, it is very undesirable to actually modify the N bytes before and 4-N bytes after the buffer. This is solved using the UAR-suffixed registers in CRYPTO when reading data out from the CRYPTO module, e.g. CRYPTO_DATA0UAR, CRYPTO_DATA1UAR, CRYPTO_DDATA0UAR,CRYPTO_DDATA1UAR, CRYPTO_QDATA0UAR, etc. When an unaligned buffer is written to a CRYPTO buffer, CRYPTO stores the N first bytes and the 4-N last bytes internally. When reading out from an UAR register, these bytes are placed back into the data if DATAxDMAPRES is set in CRYPTO_SEQCTRL.

Note that the latter case only works if the first N and the last 4-N bytes are not changed while CRYPTO works on the data. Internally CRYPTO has 2 buffers for the bytes before and after. The first one is connected to read/write of the DATA0, DDATA0 and QDATA0 registers, and the second is connected to the DATA1, DDATA1 and QDATA1 registers.

If DMAxRMODE in CRYPTO_CTRL is set to FULL or UNALIGNEDFULL and the corresponding DMAxPRES in CRYPTO_SEQCTRL is set, then a whole number of data buffers have to be written by the DMA. In all other cases, it is enough to write the number of 32-bit words to pass all LENGTH bits to the target CRYPTO buffer.

33.4.9 Debugging

There are multiple ways of debugging CRYPTO sequences. The most straight-forward way is to write individual instructions to INSTR in CRYPTO_CMD. An instruction can be written, and data can be read out and examined before running another instruction.

Running individual instructions to debug a program falls short when working with repeated sequences. In these cases, a sequence is run multiple times over a set of data. This cannot be directly replicated with individual instructions

To debug a sequence, set HALT in CRYPTO_SEQCTRL. When set, CRYPTO requires software or the debugger to step it through each instruction in the sequence. To step through the sequence, set SEQSTEP in CRYPTO_CMD. This will execute the current instruction, and make CRYPTO ready to execute the next one.

When stepping through a sequence, the current instruction index can be read from SEQIP in CRYPTO_CSTATUS. SEQSKIP, also in CRYPTO_CSTATUS tells whether the next instruction will be executed or not, based on previous conditionals in the program. SEQPART in CRYPTO_CSTATUS shows whether CRYPTO is currently in part A or B of a sequence. Even with NOBUSYSTALL in CRYPTO_CTRL cleared, read and write accesses to CRYPTO will be allowed when CRYPTO is waiting to be stepped. This is to allow data registers to be inspected during debugging.

Note: The data registers in CRYPTO (those marked read-actionable) require shifting of data in order to return the result. For this reason, reading these registers will have no effect and will return unknown values during normal debugger read accesses (see 5.3.7 Debugger Reads of Actionable Registers).

33.5 Register Map

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

Offset	Name	Type	Description
0x000	CRYPTO_CTRL	RW	Control Register
0x004	CRYPTO_WAC	RW	Wide Arithmetic Configuration
0x008	CRYPTO_CMD	W	Command Register
0x010	CRYPTO_STATUS	R	Status Register
0x014	CRYPTO_DSTATUS	R	Data Status Register
0x018	CRYPTO_CSTATUS	R	Control Status Register
0x020	CRYPTO_KEY	RWH(nB)(a)	KEY Register Access
0x024	CRYPTO_KEYBUF	RWH(nB)(a)	KEY Buffer Register Access
0x030	CRYPTO_SEQCTRL	RWH	Sequence Control
0x034	CRYPTO_SEQCTRLB	RWH	Sequence Control B
0x040	CRYPTO_IF	R	AES Interrupt Flags
0x044	CRYPTO_IFS	W1	Interrupt Flag Set Register
0x048	CRYPTO_IFC	(R)W1	Interrupt Flag Clear Register
0x04C	CRYPTO_IEN	RW	Interrupt Enable Register
0x050	CRYPTO_SEQ0	RW	Sequence Register 0
0x054	CRYPTO_SEQ1	RW	Sequence Register 1
0x058	CRYPTO_SEQ2	RW	Sequence Register 2
0x05C	CRYPTO_SEQ3	RW	Sequence Register 3
0x060	CRYPTO_SEQ4	RW	Sequence Register 4
0x080	CRYPTO_DATA0	RWH(nB)(a)	DATA0 Register Access
0x084	CRYPTO_DATA1	RWH(nB)(a)	DATA1 Register Access
0x088	CRYPTO_DATA2	RWH(nB)(a)	DATA2 Register Access
0x08C	CRYPTO_DATA3	RWH(nB)(a)	DATA3 Register Access
0x0A0	CRYPTO_DATA0XOR	RWH(nB)(a)	DATA0XOR Register Access
0x0B0	CRYPTO_DATA0BYTE	RWH(nB)(a)	DATA0 Register Byte Access
0x0B4	CRYPTO_DATA1BYTE	RWH(nB)(a)	DATA1 Register Byte Access
0x0BC	CRYPTO_DATA0XORBYTE	RWH(nB)(a)	DATA0 Register Byte XOR Access
0x0C0	CRYPTO_DATA0BYTE12	RWH(nB)	DATA0 Register Byte 12 Access
0x0C4	CRYPTO_DATA0BYTE13	RWH(nB)	DATA0 Register Byte 13 Access
0x0C8	CRYPTO_DATA0BYTE14	RWH(nB)	DATA0 Register Byte 14 Access
0x0CC	CRYPTO_DATA0BYTE15	RWH(nB)	DATA0 Register Byte 15 Access
0x100	CRYPTO_DDATA0	RWH(nB)(a)	DDATA0 Register Access
0x104	CRYPTO_DDATA1	RWH(nB)(a)	DDATA1 Register Access
0x108	CRYPTO_DDATA2	RWH(nB)(a)	DDATA2 Register Access
0x10C	CRYPTO_DDATA3	RWH(nB)(a)	DDATA3 Register Access

Offset	Name	Type	Description
0x110	CRYPTO_DDATA4	RWH(nB)(a)	DDATA4 Register Access
0x130	CRYPTO_DDATA0BIG	RWH(nB)(a)	DDATA0 Register Big Endian Access
0x140	CRYPTO_DDATA0BYTE	RWH(nB)(a)	DDATA0 Register Byte Access
0x144	CRYPTO_DDATA1BYTE	RWH(nB)(a)	DDATA1 Register Byte Access
0x148	CRYPTO_DDATA0BYTE32	RWH(nB)	DDATA0 Register Byte 32 Access
0x180	CRYPTO_QDATA0	RWH(nB)(a)	QDATA0 Register Access
0x184	CRYPTO_QDATA1	RWH(nB)(a)	QDATA1 Register Access
0x1A4	CRYPTO_QDATA1BIG	RWH(nB)(a)	QDATA1 Register Big Endian Access
0x1C0	CRYPTO_QDATA0BYTE	RWH(nB)(a)	QDATA0 Register Byte Access
0x1C4	CRYPTO_QDATA1BYTE	RWH(nB)(a)	QDATA1 Register Byte Access

33.6 Register Description

33.6.1 CRYPTO_CTRL - 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	0		0x0				0x0				0x0				0x0		0x0					0								0	0	0
Access	RW		RW				RW				RW				RW		RW					RW								RW	RW	RW
Name	COMBDMA0WEREQ		DMA1RSEL				DMA1MODE				DMA0RSEL				DMA0MODE		INCWIDTH					NOBUSYSTALL								SHA	KEYBUFDIS	AES

Bit	Name	Reset	Access	Description
31	COMBDMA0WEREQ	0	RW	**Combined Data0 Write DMA Request**
	When cleared, the DATA0WR and DATA0XORWR operate independently. When set, DATA0XORWR requests are also given through DATA0WR			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29:28	DMA1RSEL	0x0	RW	**DATA0 DMA Unaligned Read Register Select**
	Specifies which read register is used for DMA1RD DMA requests (see related notes in 33.4.8 DMA and 33.4.3 Repeated Sequence)			

Value	Mode	Description
0	DATA1	
1	DDATA1	
2	QDATA1	
3	QDATA1BIG	

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	DMA1MODE	0x0	RW	**DMA1 Read Mode**
	This field determines how data is read when using DMA			

Value	Mode	Description
0	FULL	Target register is fully read/written during every DMA transaction
1	LENLIMIT	Length Limited. When the current length, i.e. LENGTHA or LENGTHB indicates that there are less bytes available than the register size, only length + 1 bytes + necessary zero padding is read. Zero padding is automatically added when writing.
2	FULLBYTE	Target register is fully read/written during every DMA transaction. Bytewise DMA.

Bit	Name	Reset	Access	Description
	3	LENLIMITBYTE		Length Limited. When the current length, i.e. LENGTHA or LENGTHB indicates that there are less bytes available than the register size, only length + 1 bytes + necessary zero padding is read. Bytewise DMA. Zero padding is automatically added when writing.
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	DMA0RSEL	0x0	RW	**DMA0 Read Register Select**
	Specifies which read register is used for DMA0RD DMA requests (see related notes in 33.4.8 DMA and 33.4.3 Repeated Sequence)			
	Value	Mode		Description
	0	DATA0		
	1	DDATA0		
	2	DDATA0BIG		
	3	QDATA0		
19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	DMA0MODE	0x0	RW	**DMA0 Read Mode**
	This field determines how data is read when using DMA.			
	Value	Mode		Description
	0	FULL		Target register is fully read/written during every DMA transaction
	1	LENLIMIT		Length Limited. When the current length, i.e. LENGTHA or LENGTHB indicates that there are less bytes available than the register size, only length + necessary zero padding is read. Zero padding is automatically added when writing.
	2	FULLBYTE		Target register is fully read/written during every DMA transaction. Bytewise DMA.
	3	LENLIMITBYTE		Length Limited. When the current length, i.e. LENGTHA or LENGTHB indicates that there are less bytes available than the register size, only length + necessary zero padding is read. Bytewise DMA. Zero padding is automatically added when writing.
15:14	INCWIDTH	0x0	RW	**Increment Width**
	This field determines the number of bytes used for the increment function in data1.			
	Value	Mode		Description
	0	INCWIDTH1		Byte 15 in DATA1 is used for the increment function.
	1	INCWIDTH2		Bytes 14 and 15 in DATA1 are used for the increment function.
	2	INCWIDTH3		Bytes 13 to 15 in DATA1 are used for the increment function.
	3	INCWIDTH4		Bytes 12 to 15 in DATA1 are used for the increment function.
13:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	NOBUSYSTALL	0	RW	**No Stalling of Bus When Busy**
	When set, bus accesses will not be stalled on access during an operation			

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	SHA	0	RW	**SHA Mode**
	Select SHA-1 or SHA-2 mode.			
	Value	Mode		Description
	0	SHA1		SHA-1 mode
	1	SHA2		SHA-2 mode (SHA-224 or SHA-256)
1	KEYBUFDIS	0	RW	**Key Buffer Disable**
	Set to Disable key buffering.			
0	AES	0	RW	**AES Mode**
	Select AES mode			
	Value	Mode		Description
	0	AES128		AES-128 mode
	1	AES256		AES-256 mode

33.6.2 CRYPTO_WAC - Wide Arithmetic Configuration

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					0		0x0		
Access																					RW		RW					RW		RW		
Name																					RESULTWIDTH		MULWIDTH					MODOP		MODULUS		

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	RESULTWIDTH	0x0	RW	**Result Width**

Result-size for non-modulus instructions

Value	Mode	Description
0	256BIT	Results have 256 bits
1	128BIT	Results have 128 bits
2	260BIT	Results have 260 bits. Upper bits of result can be read through DDATA0MSBS in CRYPTO_STATUS

Bit	Name	Reset	Access	Description
9:8	MULWIDTH	0x0	RW	**Multiply Width**

Number of bits to multiply on non-modulus multiply instruction

Value	Mode	Description
0	MUL256	Multiply 256 bits
1	MUL128	Multiply 128 bits
2	MULMOD	Same number of bits as specified by MODULUS

Bit	Name	Reset	Access	Description
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	MODOP	0	RW	**Modular Operation Field Type**

Field type used for modular operations

Value	Mode	Description
0	BINARY	Modular operations use XOR as required by certain algorithms
1	REGULAR	Modular operations use normal modular arithmetic, not XOR

Bit	Name	Reset	Access	Description
3:0	MODULUS	0x0	RW	**Modular Operation Modulus**

Modulus used for modular operations

Value	Mode	Description
0	BIN256	Generic modulus. p = 2^256

Bit	Name	Reset	Access	Description
	1	BIN128		Generic modulus. p = 2^128
	2	ECCBIN233P		Modulus for B-233 and K-233 ECC curves. p(t) = t^233 + t^74 + 1
	3	ECCBIN163P		Modulus for B-163 and K-163 ECC curves. p(t) = t^163 + t^7 + t^6 + t^3 + 1
	4	GCMBIN128		Modulus for GCM. P(t) = t^128 + t^7 + t^2 + t + 1
	5	ECCPRIME256P		Modulus for P-256 ECC curve. p = 2^256 - 2^224 + 2^192 + 2^96 - 1
	6	ECCPRIME224P		Modulus for P-224 ECC curve. p = 2^224 - 2^96 - 1
	7	ECCPRIME192P		Modulus for P-192 ECC curve. p = 2^192 - 2^64 - 1
	8	ECCBIN233N		P modulus for B-233 ECC curve
	9	ECCBIN233KN		P modulus for K-233 ECC curve
	10	ECCBIN163N		P modulus for B-163 ECC curve
	11	ECCBIN163KN		P modulus for K-163 ECC curve
	12	ECCPRIME256N		P modulus for P-256 ECC curve
	13	ECCPRIME224N		P modulus for P-224 ECC curve
	14	ECCPRIME192N		P modulus for P-192 ECC curve

33.6.3 CRYPTO_CMD - Command 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																					0	0	0		0x00							
Access																					W1	W1	W1		W							
Name																					SEQSTEP	SEQSTOP	SEQSTART		INSTR							

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	SEQSTEP	0	W1	**Sequence Step** When in a halted sequence, executes the current instruction and moves to the next
10	SEQSTOP	0	W1	**Sequence Stop** Set to stop encryption/decryption regardless of it being a single or a SEQUENCE.
9	SEQSTART	0	W1	**Encryption/Decryption SEQUENCE Start** Set to start encryption/decryption SEQUENCE.
8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	INSTR	0x00	W	**Execute Instruction** Write to this field to perform any of the instructions described below. Illegal values are ignored. See 33.4.2.2 Available Instructions for details and requirements of each instruction

Value	Mode	Description
0	END	End of program
1	EXEC	Start executing instructions up to this point, which also marks end of program
3	DATA1INC	See detailed instruction listing
4	DATA1INCCLR	See detailed instruction listing
5	AESENC	AES Encryption
6	AESDEC	AES Decryption
7	SHA	SHA
8	ADD	Add
9	ADDC	Add with carry
10	LADD	Long addition
11	LADDC	Long addition with carry
12	MADD	Modular addition
13	MADD32	Word-wise addition
16	SUB	Subtract

Bit	Name	Reset	Access	Description
	17	SUBC		Subtract with carry
	18	LSUB		Long subtraction
	19	LSUBC		Long subtract with carry
	20	MSUB		Modular subtraction
	24	MUL		Multiply
	25	MULC		See detailed instruction listing
	26	LMUL		Long multiply
	28	MMUL		Modular multiplication
	29	MULO		See detailed instruction listing
	31	LMULO		See detailed instruction listing
	32	SHL		Shift left
	33	SHLC		Shift left with carry (Rotate left)
	34	SHLB		See detailed instruction listing
	35	SHL1		See detailed instruction listing
	36	SHR		Shift right
	37	SHRC		Shift right with carry (Rotate right)
	38	SHRB		See detailed instruction listing
	39	SHR1		See detailed instruction listing
	40	ADDO		See detailed instruction listing
	41	ADDIC		See detailed instruction listing
	42	LADDO		See detailed instruction listing
	43	LADDIC		See detailed instruction listing
	48	CLR		Clear DDATA0
	49	XOR		XOR
	50	INV		Invert operand
	52	CSET		Carry set
	53	CCLR		Carry clear
	54	BBSWAP128		See detailed instruction listing
	56	INC		Increment DDATA0
	57	DEC		Decrement DDATA0
	58	LINC		Long increment
	59	LDEC		Long decrement
	62	SHRA		Arithmetic shift right
	64	DATA0TODATA0		DATA0 = DATA0
	65	DATA0TODATA0XOR		DATA0 = DATA0 ^ DATA0
	66	DATA0TODATA0XOR-LEN		DATA0[len-1:0] = DATA0[len-1:0] ^ DATA0[len-1:0]

Bit	Name	Reset	Access	Description
	68	DATA0TODATA1		DATA1 = DATA0
	69	DATA0TODATA2		DATA2 = DATA0
	70	DATA0TODATA3		DATA3 = DATA0
	72	DATA1TODATA0		DATA0 = DATA1
	73	DATA1TODATA0XOR		DATA0 = DATA0 ^ DATA1
	74	DATA1TODATA0XOR-LEN		DATA0[len-1:0] = DATA0[len-1:0] ^ DATA1[len-1:0]
	77	DATA1TODATA2		DATA2 = DATA1
	78	DATA1TODATA3		DATA3 = DATA1
	80	DATA2TODATA0		DATA0 = DATA2
	81	DATA2TODATA0XOR		DATA0 = DATA0 ^ DATA2
	82	DATA2TODATA0XOR-LEN		DATA0[len-1:0] = DATA0[len-1:0] ^ DATA2[len-1:0]
	84	DATA2TODATA1		DATA1 = DATA2
	86	DATA2TODATA3		DATA3 = DATA2
	88	DATA3TODATA0		DATA0 = DATA3
	89	DATA3TODATA0XOR		DATA0 = DATA0 ^ DATA3
	90	DATA3TODATA0XOR-LEN		DATA0[len-1:0] = DATA0[len-1:0] ^ DATA3[len-1:0]
	92	DATA3TODATA1		DATA1 = DATA3
	93	DATA3TODATA2		DATA2 = DATA3
	99	DATATODMA0		See detailed instruction listing
	107	DATATODMA1		See detailed instruction listing
	112	DMA0TODATA		See detailed instruction listing
	113	DMA0TODATAXOR		See detailed instruction listing
	114	DMA1TODATA		See detailed instruction listing
	129	DDATA0TODDATA1		DDATA1 = DDATA0
	130	DDATA0TODDATA2		DDATA2 = DDATA0
	131	DDATA0TODDATA3		DDATA3 = DDATA0
	132	DDATA0TODDATA4		DDATA4 = DDATA0
	133	DDATA0LTODATA0		DATA0 = DDATA0[127:0]
	134	DDATA0HTODATA1		DATA1 = DDATA0[255:128]
	135	DDATA0LTODATA2		DATA2 = DDATA0[127:0]
	136	DDATA1TODDATA0		DDATA0 = DDATA1
	138	DDATA1TODDATA2		DDATA2 = DDATA1
	139	DDATA1TODDATA3		DDATA3 = DDATA1
	140	DDATA1TODDATA4		DDATA4 = DDATA1
	141	DDATA1LTODATA0		DATA0 = DDATA1[127:0]

Bit	Name	Reset	Access	Description
142	DDATA1HTODATA1			DATA1 = DDATA1[255:128]
143	DDATA1LTODATA2			DATA2 = DDATA1[127:0]
144	DDATA2TODDATA0			DDATA0 = DDATA2
145	DDATA2TODDATA1			DDATA1 = DDATA2
147	DDATA2TODDATA3			DDATA3 = DDATA2
148	DDATA2TODDATA4			DDATA4 = DDATA2
151	DDATA2LTODATA2			DATA2 = DDATA2[127:0]
152	DDATA3TODDATA0			DDATA0 = DDATA3
153	DDATA3TODDATA1			DDATA1 = DDATA3
154	DDATA3TODDATA2			DDATA2 = DDATA3
156	DDATA3TODDATA4			DDATA4 = DDATA3
157	DDATA3LTODATA0			DATA0 = DDATA3[127:0]
158	DDATA3HTODATA1			DATA1 = DDATA3[255:128]
160	DDATA4TODDATA0			DDATA0 = DDATA4
161	DDATA4TODDATA1			DDATA1 = DDATA4
162	DDATA4TODDATA2			DDATA2 = DDATA4
163	DDATA4TODDATA3			DDATA3 = DDATA4
165	DDATA4LTODATA0			DATA0 = DDATA4[127:0]
166	DDATA4HTODATA1			DATA1 = DDATA4[255:128]
167	DDATA4LTODATA2			DATA2 = DDATA4[127:0]
168	DATA0TODDATA0			DDATA0 = DATA0
169	DATA0TODDATA1			DDATA1 = DATA0
176	DATA1TODDATA0			DDATA0 = DATA1
177	DATA1TODDATA1			DDATA1 = DATA1
184	DATA2TODDATA0			DDATA0 = DATA2
185	DATA2TODDATA1			DDATA1 = DATA2
186	DATA2TODDATA2			DDATA2 = DATA2
192	SELDDATA0DDATA0			Use DDATA0 as V0, DDATA0 as V1
193	SELDDATA1DDATA0			Use DDATA1 as V0, DDATA0 as V1
194	SELDDATA2DDATA0			Use DDATA2 as V0, DDATA0 as V1
195	SELDDATA3DDATA0			Use DDATA3 as V0, DDATA0 as V1
196	SELDDATA4DDATA0			Use DDATA4 as V0, DDATA0 as V1
197	SELDATA0DDATA0			Use DATA0 as V0, DDATA0 as V1
198	SELDATA1DDATA0			Use DATA1 as V0, DDATA1 as V1
199	SELDATA2DDATA0			Use DATA2 as V0, DDATA2 as V1
200	SELDDATA0DDATA1			Use DDATA0 as V0, DDATA1 as V1
201	SELDDATA1DDATA1			Use DDATA1 as V0, DDATA1 as V1

Bit	Name	Reset	Access	Description
	202	SELDDATA2DDATA1		Use DDATA2 as V0, DDATA1 as V1
	203	SELDDATA3DDATA1		Use DDATA3 as V0, DDATA1 as V1
	204	SELDDATA4DDATA1		Use DDATA4 as V0, DDATA1 as V1
	205	SELDATA0DDATA1		Use DATA0 as V0, DDATA0 as V1
	206	SELDATA1DDATA1		Use DATA1 as V0, DDATA1 as V1
	207	SELDATA2DDATA1		Use DATA2 as V0, DDATA2 as V1
	208	SELDDATA0DDATA2		Use DDATA0 as V0, DDATA2 as V1
	209	SELDDATA1DDATA2		Use DDATA1 as V0, DDATA2 as V1
	210	SELDDATA2DDATA2		Use DDATA2 as V0, DDATA2 as V1
	211	SELDDATA3DDATA2		Use DDATA3 as V0, DDATA2 as V1
	212	SELDDATA4DDATA2		Use DDATA4 as V0, DDATA2 as V1
	213	SELDATA0DDATA2		Use DATA0 as V0, DDATA0 as V1
	214	SELDATA1DDATA2		Use DATA1 as V0, DDATA1 as V1
	215	SELDATA2DDATA2		Use DATA2 as V0, DDATA2 as V1
	216	SELDDATA0DDATA3		Use DDATA0 as V0, DDATA3 as V1
	217	SELDDATA1DDATA3		Use DDATA1 as V0, DDATA3 as V1
	218	SELDDATA2DDATA3		Use DDATA2 as V0, DDATA3 as V1
	219	SELDDATA3DDATA3		Use DDATA3 as V0, DDATA3 as V1
	220	SELDDATA4DDATA3		Use DDATA4 as V0, DDATA3 as V1
	221	SELDATA0DDATA3		Use DATA0 as V0, DDATA0 as V1
	222	SELDATA1DDATA3		Use DATA1 as V0, DDATA1 as V1
	223	SELDATA2DDATA3		Use DATA2 as V0, DDATA2 as V1
	224	SELDDATA0DDATA4		Use DDATA0 as V0, DDATA4 as V1
	225	SELDDATA1DDATA4		Use DDATA1 as V0, DDATA4 as V1
	226	SELDDATA2DDATA4		Use DDATA2 as V0, DDATA4 as V1
	227	SELDDATA3DDATA4		Use DDATA3 as V0, DDATA4 as V1
	228	SELDDATA4DDATA4		Use DDATA4 as V0, DDATA4 as V1
	229	SELDATA0DDATA4		Use DATA0 as V0, DDATA4 as V1
	230	SELDATA1DDATA4		Use DATA1 as V0, DDATA4 as V1
	231	SELDATA2DDATA4		Use DATA2 as V0, DDATA4 as V1
	232	SELDDATA0DATA0		Use DDATA0 as V0, DATA0 as V1
	233	SELDDATA1DATA0		Use DDATA1 as V0, DATA0 as V1
	234	SELDDATA2DATA0		Use DDATA2 as V0, DATA0 as V1
	235	SELDDATA3DATA0		Use DDATA3 as V0, DATA0 as V1
	236	SELDDATA4DATA0		Use DDATA4 as V0, DATA0 as V1
	237	SELDATA0DATA0		Use DATA0 as V0, DATA0 as V1
	238	SELDATA1DATA0		Use DATA1 as V0, DATA0 as V1

Bit	Name	Reset	Access	Description
	239	SELDATA2DATA0		Use DATA2 as V0, DATA0 as V1
	240	SELDDATA0DATA1		Use DDATA0 as V0, DATA1 as V1
	241	SELDDATA1DATA1		Use DDATA1 as V0, DATA1 as V1
	242	SELDDATA2DATA1		Use DDATA2 as V0, DATA1 as V1
	243	SELDDATA3DATA1		Use DDATA3 as V0, DATA1 as V1
	244	SELDDATA4DATA1		Use DDATA4 as V0, DATA1 as V1
	245	SELDATA0DATA1		Use DATA0 as V0, DATA1 as V1
	246	SELDATA1DATA1		Use DATA1 as V0, DATA1 as V1
	247	SELDATA2DATA1		Use DATA2 as V0, DATA1 as V1
	248	EXECIFA		Run following if in A sequence
	249	EXECIFB		Run following if in B sequence
	250	EXECIFNLAST		Run following if in last iteration of combined A and B sequence
	251	EXECIFLAST		Run following if in last iteration of combined A and B sequence
	252	EXECIFCARRY		Run following if CARRY bit is set
	253	EXECIFNCARRY		Run following if CARRY bit is not set
	254	EXECALWAYS		Resume execution

33.6.4 CRYPTO_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																														0	0	0
Access																														R	R	R
Name																														DMAACTIVE	INSTRRUNNING	SEQRUNNING

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	DMAACTIVE	0	R	**DMA Action is Active**
	This bit indicates that the AES module is waiting for a DMA transfer to complete.			
1	INSTRRUNNING	0	R	**Action is Active**
	This bit indicates that the AES module busy executing an instruction. The origin of the instruction is either through CRYPTO_CMD or due to a running SEQUENCE.			
0	SEQRUNNING	0	R	**AES SEQUENCE Running**
	This bit indicates that the AES module is running an encryption/decryption SEQUENCE.			

33.6.5 CRYPTO_DSTATUS - Data 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								0				X	0xX								0xX								0xX			
Access								R				R	R								R								R			
Name								CARRY				DDATA1MSB	DDATA0MSBS								DDATA0LSBS								DATA0ZERO			

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	CARRY	0	R	**Carry From Arithmetic Operation**
	Set on carry from arithmetic operations			
23:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	DDATA1MSB	X	R	**MSB in DDATA1**
	Allows read of 255 in DDATA1. Does not depend on RESULTWIDTH in CRYPTO_WAC			
19:16	DDATA0MSBS	0xX	R	**MSB in DDATA0**
	Allows read of 4 MSBs in DDATA0. The bits depend on RESULTWIDTH in CRYPTO_WAC			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	DDATA0LSBS	0xX	R	**LSBs in DDATA0**
	Allows read of 4 LSBs in DDATA0			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	DATA0ZERO	0xX	R	**Data 0 Zero**
	This field contains flags indicating if any 32 bit part of DATA0 is 0.			

Value	Mode	Description
1	ZERO0TO31	In DATA0 bits 0 to 31 are all zero.
2	ZERO32TO63	In DATA0 bits 32 to 63 are all zero.
4	ZERO64TO95	In DATA0 bits 64 to 95 are all zero.
8	ZERO96TO127	In DATA0 bits 96 to 127 are all zero.

33.6.6 CRYPTO_CSTATUS - Control 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										0x00					0	0							0x2								0x1	
Access										R					R	R							R								R	
Name										SEQIP					SEQSKIP	SEQPART							V1								V0	

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*		
24:20	SEQIP	0x00	R	**Sequence Next Instruction Pointer**
	Next sequence instruction when in halted sequence			
19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17	SEQSKIP	0	R	**Sequence Skip Next Instruction**
	When in halted sequence, tells whether next instruction will be skipped			
16	SEQPART	0	R	**Sequence Part**
	Shows whether currently in part A or B of a sequence			

Value	Mode	Description
0	SEQA	
1	SEQB	

Bit	Name	Reset	Access	Description
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	V1	0x2	R	**Selected ALU Operand 1**
	Selectable operand for arithmetic operations			

Value	Mode	Description
0	DDATA0	
1	DDATA1	
2	DDATA2	
3	DDATA3	
4	DDATA4	
5	DATA0	
6	DATA1	
7	DATA2	

Bit	Name	Reset	Access	Description
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
2:0	V0	0x1	R	**Selected ALU Operand 0**

Selectable operand for arithmetic operations

Value	Mode	Description
0	DDATA0	
1	DDATA1	
2	DDATA2	
3	DDATA3	
4	DDATA4	
5	DATA0	
6	DATA1	
7	DATA2	

33.6.7 CRYPTO_KEY - KEY Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	KEY																															

Bit	Name	Reset	Access	Description
31:0	KEY	0xXXXXXXXX	RWH	**Key Access**

Access the KEY. 4x32bits (8x32bits if AES256 in CRYPTO_CTRL is set) read/write accesses are required to fully read/write KEY.

33.6.8 CRYPTO_KEYBUF - KEY Buffer Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	KEYBUF																															

Bit	Name	Reset	Access	Description
31:0	KEYBUF	0xXXXXXXXX	RWH	**Key Buffer Access**
	Access to KEYBUF. 4x32bits (8x32bits if AES256 in CRYPTO_CTRL is set) read/write accesses are required to fully read/write KEYBUF			

33.6.9 CRYPTO_SEQCTRL - Sequence 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	0		0	0	0x0		0x0				0x0								0x0000													
Access	RW		RW	RW	RWH		RWH				RW								RWH													
Name	HALT		DMA1PRESA	DMA0PRESA	DMA1SKIP		DMA0SKIP				BLOCKSIZE								LENGTHA													

Bit	Name	Reset	Access	Description
31	HALT	0	RW	**Halt Sequence**
	Allows stepping through CRYPTO instructions in the sequence for debugging.			
30	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
29	DMA1PRESA	0	RW	**DMA1 Preserve a**
	Set to write skipped bytes back on next DMA1WR triggered write. Use this together with DMA1SKIP to enable in-place conversions with CRYPTO			
28	DMA0PRESA	0	RW	**DMA0 Preserve a**
	Set to write skipped bytes back on next DMA0WR triggered write. Use this together with DMA0SKIP to enable in-place conversions with CRYPTO			
27:26	DMA1SKIP	0x0	RWH	**DMA1 Skip**
	Set to number of bytes to exclude from data received by next DMA1RD insruction			
25:24	DMA0SKIP	0x0	RWH	**DMA0 Skip**
	Set to number of bytes to exclude from data received by next DMA0RD insruction			
23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	BLOCKSIZE	0x0	RW	**Size of Data Blocks**
	Defines the width of blocks processed in each iteration of a sequence running on a dataset (see related note in 33.4.3 Repeated Sequence)			

Value	Mode	Description
0	16BYTES	A block is 16 bytes long
1	32BYTES	A block is 32 bytes long
2	64BYTES	A block is 64 bytes long

19:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:0	LENGTHA	0x0000	RWH	**Buffer Length a in Bytes**
	This field sets the number of bytes to be handled during the repeated sequence. Set it to the exact number of bytes. If the number is not a multiple of BLOCKSIZE, the last data block is zero-padded. Format is unsigned integer.			

33.6.10 CRYPTO_SEQCTRLB - Sequence Control B

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			0	0																					0x0000						
Access			RW	RW																					RWH						
Name			DMA1PRESB	DMA0PRESB																					LENGTHB						

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	DMA1PRESB	0	RW	**DMA1 Preserve B**
	For unaligned sequences, set this bit along with DMA1PRESA for in-place conversions where all data is written out from CRYPTO again. If only the second part of a data-set is written, enable only this to preserve the data read in during part A			
28	DMA0PRESB	0	RW	**DMA0 Preserve B**
	For unaligned sequences, set this bit along with DMA0PRESA for in-place conversions where all data is written out from CRYPTO again. If only the second part of a data-set is written, enable only this to preserve the data read in during part A			
27:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:0	LENGTHB	0x0000	RWH	**Buffer Length B in Bytes**
	Sets the number of bytes to be handled in a second iteration over a programmed sequence.			

33.6.11 CRYPTO_IF - AES Interrupt Flags

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																															0	0
Access																															R	R
Name																															SEQDONE	INSTRDONE

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	SEQDONE	0	R	**Sequence Done**
	Set when an instruction sequence has completed			
0	INSTRDONE	0	R	**Instruction Done**
	Set when an instruction has completed			

33.6.12 CRYPTO_IFS - Interrupt Flag Set 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																															0	0
Access																															W1	W1
Name																															SEQDONE	INSTRDONE

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	SEQDONE	0	W1	**Set SEQDONE Interrupt Flag**
	Write 1 to set the SEQDONE interrupt flag			
0	INSTRDONE	0	W1	**Set INSTRDONE Interrupt Flag**
	Write 1 to set the INSTRDONE interrupt flag			

33.6.13 CRYPTO_IFC - Interrupt Flag Clear 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																															0	0
Access																															(R)W1	(R)W1
Name																															SEQDONE	INSTRDONE

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	SEQDONE	0	(R)W1	**Clear SEQDONE Interrupt Flag**
	Write 1 to clear the SEQDONE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	INSTRDONE	0	(R)W1	**Clear INSTRDONE Interrupt Flag**
	Write 1 to clear the INSTRDONE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

33.6.14 CRYPTO_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																															0	0
Access																															RW	RW
Name																															SEQDONE	INSTRDONE

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	SEQDONE	0	RW	**SEQDONE Interrupt Enable**
	Enable/disable the SEQDONE interrupt			
0	INSTRDONE	0	RW	**INSTRDONE Interrupt Enable**
	Enable/disable the INSTRDONE interrupt			

33.6.15 CRYPTO_SEQ0 - Sequence Register 0

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	0x00								0x00								0x00								0x00							
Access	RW								RW								RW								RW							
Name	INSTR3								INSTR2								INSTR1								INSTR0							

Bit	Name	Reset	Access	Description
31:24	INSTR3	0x00	RW	**Sequence Instruction 3**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
23:16	INSTR2	0x00	RW	**Sequence Instruction 2**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
15:8	INSTR1	0x00	RW	**Sequence Instruction 1**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
7:0	INSTR0	0x00	RW	**Sequence Instruction 0**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			

33.6.16 CRYPTO_SEQ1 - Sequence Register 1

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	0x00								0x00								0x00								0x00							
Access	RW								RW								RW								RW							
Name	INSTR7								INSTR6								INSTR5								INSTR4							

Bit	Name	Reset	Access	Description
31:24	INSTR7	0x00	RW	**Sequence Instruction 7**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
23:16	INSTR6	0x00	RW	**Sequence Instruction 6**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
15:8	INSTR5	0x00	RW	**Sequence Instruction 5**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
7:0	INSTR4	0x00	RW	**Sequence Instruction 4**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			

33.6.17 CRYPTO_SEQ2 - Sequence Register 2

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	0x00								0x00								0x00								0x00							
Access	RW								RW								RW								RW							
Name	INSTR11								INSTR10								INSTR9								INSTR8							

Bit	Name	Reset	Access	Description
31:24	INSTR11	0x00	RW	**Sequence Instruction 11**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
23:16	INSTR10	0x00	RW	**Sequence Instruction 10**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
15:8	INSTR9	0x00	RW	**Sequence Instruction 9**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
7:0	INSTR8	0x00	RW	**Sequence Instruction 8**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			

33.6.18 CRYPTO_SEQ3 - Sequence Register 3

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	0x00								0x00								0x00								0x00							
Access	RW								RW								RW								RW							
Name	INSTR15								INSTR14								INSTR13								INSTR12							

Bit	Name	Reset	Access	Description
31:24	INSTR15	0x00	RW	**Sequence Instruction 15**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
23:16	INSTR14	0x00	RW	**Sequence Instruction 14**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
15:8	INSTR13	0x00	RW	**Sequence Instruction 13**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
7:0	INSTR12	0x00	RW	**Sequence Instruction 12**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			

33.6.19 CRYPTO_SEQ4 - Sequence Register 4

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	0x00								0x00								0x00								0x00							
Access	RW								RW								RW								RW							
Name	INSTR19								INSTR18								INSTR17								INSTR16							

Bit	Name	Reset	Access	Description
31:24	INSTR19	0x00	RW	**Sequence Instruction 19**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
23:16	INSTR18	0x00	RW	**Sequence Instruction 18**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
15:8	INSTR17	0x00	RW	**Sequence Instruction 17**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			
7:0	INSTR16	0x00	RW	**Sequence Instruction 16**
	Sequence instruction. See INSTR in CRYPTO_CMD for a possible values.			

33.6.20 CRYPTO_DATA0 - DATA0 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DATA0																															

Bit	Name	Reset	Access	Description
31:0	DATA0	0xXXXXXXXX	RWH	**Data 0 Access**
	Access to DATA0. 4x32bits read/write accesses are required to fully read/write DATA0			

33.6.21 CRYPTO_DATA1 - DATA1 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DATA1																															

Bit	Name	Reset	Access	Description
31:0	DATA1	0xXXXXXXXX	RWH	**Data 1 Access**
	Access to DATA1. 4x32bits read/write accesses are required to fully read/write DATA1			

33.6.22 CRYPTO_DATA2 - DATA2 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DATA2																															

Bit	Name	Reset	Access	Description
31:0	DATA2	0xXXXXXXXX	RWH	**Data 2 Access**
	Access to DATA2. 4x32bits read/write accesses are required to fully read/write DATA2.			

33.6.23 CRYPTO_DATA3 - DATA3 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DATA3																															

Bit	Name	Reset	Access	Description
31:0	DATA3	0xXXXXXXXX	RWH	**Data 3 Access**
	Access to DATA3. 4x32bits read/write accesses are required to fully read/write DATA3.			

33.6.24 CRYPTO_DATA0XOR - DATA0XOR Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DATA0XOR																															

Bit	Name	Reset	Access	Description
31:0	DATA0XOR	0xXXXXXXXX	RWH	**XOR Data 0 Access**

Any value written to this register will be XOR'ed with the value of DATA0. The result is stored in DATA0. Reads return DATA0 directly. 4x32bits read/write accesses are required to perform a full XOR write to DATA0

33.6.25 CRYPTO_DATA0BYTE - DATA0 Register Byte Access (No Bit Access) (Actionable Reads)

Offset	Bit Position																															
0x0B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0xXX							
Access																									RWH							
Name																									DATA0BYTE							

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	DATA0BYTE	0xXX	RWH	**Data 0 Byte Access**

Access to DATA0. 16x8bits read/write accesses are required to fully read/write DATA0. Accesses must be performed in multiples of 4, or data incoherency may occur

33.6.26 CRYPTO_DATA1BYTE - DATA1 Register Byte Access (No Bit Access) (Actionable Reads)

Offset	Bit Position																													
0x0B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0xXX							
Access																									RWH							
Name																									DATA1BYTE							

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	DATA1BYTE	0xXX	RWH	**Data 1 Byte Access**
	Access to DATA1. 16x8bits read/write accesses are required to fully read/write DATA1. Accesses must be performed in multiples of 4, or data incoherency may occur			

33.6.27 CRYPTO_DATA0XORBYTE - DATA0 Register Byte XOR Access (No Bit Access) (Actionable Reads)

Offset	Bit Position																													
0x0BC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0xXX							
Access																									RWH							
Name																									DATA0XORBYTE							

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	DATA0XORBYTE	0xXX	RWH	**Data 0 XOR Byte Access**
	Access to DATA0. 16x8bits read/write accesses are required to fully read/write DATA0. Written data is XOR'ed with the already present data in DATA0. Accesses must be performed in multiples of 4, or data incoherency may occur			

33.6.28 CRYPTO_DATA0BYTE12 - DATA0 Register Byte 12 Access (No Bit Access)

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																									0xXX							
Access																									RWH							
Name																									DATA0BYTE12							

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	DATA0BYTE12	0xXX	RWH	**Data 0 Byte 12 Access**
	Access to DATA0 byte 12.			

33.6.29 CRYPTO_DATA0BYTE13 - DATA0 Register Byte 13 Access (No Bit Access)

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																									0xXX							
Access																									RWH							
Name																									DATA0BYTE13							

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	DATA0BYTE13	0xXX	RWH	**Data 0 Byte 13 Access**
	Access to DATA0 byte 13.			

33.6.30 CRYPTO_DATA0BYTE14 - DATA0 Register Byte 14 Access (No Bit Access)

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																									0xXX							
Access																									RWH							
Name																									DATA0BYTE14							

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	DATA0BYTE14	0xXX	RWH	**Data 0 Byte 14 Access**
	Access to DATA0 byte 14.			

33.6.31 CRYPTO_DATA0BYTE15 - DATA0 Register Byte 15 Access (No Bit Access)

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																									0xXX							
Access																									RWH							
Name																									DATA0BYTE15							

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	DATA0BYTE15	0xXX	RWH	**Data 0 Byte 15 Access**
	Access to DATA0 byte 15.			

33.6.32 CRYPTO_DDATA0 - DDATA0 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DDATA0																															

Bit	Name	Reset	Access	Description
31:0	DDATA0	0xXXXXXXXX	RWH	**Double Data 0 Access**
	Access to DDATA0. 8x32bits read/write accesses are required to fully read/write DDATA0.			

33.6.33 CRYPTO_DDATA1 - DDATA1 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DDATA1																															

Bit	Name	Reset	Access	Description
31:0	DDATA1	0xXXXXXXXX	RWH	**Double Data 0 Access**
	Access to DDATA1, which is equal to the full width of KEY regardless of AES256 in CRYPTO_CTRL. 8x32bits read/write accesses are required to fully read/write DDATA1.			

33.6.34 CRYPTO_DDATA2 - DDATA2 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DDATA2																															

Bit	Name	Reset	Access	Description
31:0	DDATA2	0xXXXXXXXX	RWH	**Double Data 0 Access**

Access to DDATA2, which consists of {DATA1, DATA0}. 8x32bits read/write accesses are required to fully read/write DDATA2.

33.6.35 CRYPTO_DDATA3 - DDATA3 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DDATA3																															

Bit	Name	Reset	Access	Description
31:0	DDATA3	0xXXXXXXXX	RWH	**Double Data 0 Access**

Access to DDATA3, which consists of {DATA3, DATA2}. 8x32bits read/write accesses are required to fully read/write DDATA3.

33.6.36 CRYPTO_DDATA4 - DDATA4 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DDATA4																															

Bit	Name	Reset	Access	Description
31:0	DDATA4	0xXXXXXXXX	RWH	**Double Data 0 Access**

Access to DDATA4, which is equal to the full width of KEYBUF regardless of AES256 in CRYPTO_CTRL. 8x32bits read/write accesses are required to fully read/write DDATA4.

33.6.37 CRYPTO_DDATA0BIG - DDATA0 Register Big Endian Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	DDATA0BIG																															

Bit	Name	Reset	Access	Description
31:0	DDATA0BIG	0xXXXXXXXX	RWH	**Double Data 0 Big Endian Access**

Big endian access to DDATA0. 8x32bits read/write accesses are required to fully read/write DDATA0.

33.6.38 CRYPTO_DDATA0BYTE - DDATA0 Register Byte Access (No Bit Access) (Actionable Reads)

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																									0xXX							
Access																									RWH							
Name																									DDATA0BYTE							

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	DDATA0BYTE	0xXX	RWH	**Ddata 0 Byte Access**
		Access to DDATA0. 32x8bits read/write accesses are required to fully read/write DDATA0. Accesses must be performed in multiples of 4, or data incoherency may occur		

33.6.39 CRYPTO_DDATA1BYTE - DDATA1 Register Byte Access (No Bit Access) (Actionable Reads)

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																									0xXX							
Access																									RWH							
Name																									DDATA1BYTE							

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	DDATA1BYTE	0xXX	RWH	**Ddata 1 Byte Access**
		Access to DDATA1. 32x8bits read/write accesses are required to fully read/write DDATA1. Accesses must be performed in multiples of 4, or data incoherency may occur		

33.6.40 CRYPTO_DDATA0BYTE32 - DDATA0 Register Byte 32 Access (No Bit Access)

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																													0xX				
Access																													RWH				
Name																													DDATA0BYTE32				

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	DDATA0BYTE32	0xX	RWH	**Ddata 0 Byte 32 Access**
	Access to DDATA0 byte 32. This is used when RESULTWIDTH in CRYPTO_WAC is set to 260BIT.			

33.6.41 CRYPTO_QDATA0 - QDATA0 Register Access (No Bit Access) (Actionable Reads)

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																0xXXXXXXXX																
Access																RWH																
Name																QDATA0																

Bit	Name	Reset	Access	Description
31:0	QDATA0	0xXXXXXXXX	RWH	**Quad Data 0 Access**
	Access to QDATA0, which is equal to {DDATA1, DDATA0}. 16x32bits read/write accesses are required to fully read/write QDATA0.			

33.6.42 CRYPTO_QDATA1 - QDATA1 Register Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	QDATA1																															

Bit	Name	Reset	Access	Description
31:0	QDATA1	0xXXXXXXXX	RWH	**Quad Data 1 Access**

Access to QDATA1, which is equal to {DATA3, DATA2, DATA1, DATA0} and {DDATA3, DDATA2}. 16x32bits read/write accesses are required to fully read/write QDATA1.

33.6.43 CRYPTO_QDATA1BIG - QDATA1 Register Big Endian Access (No Bit Access) (Actionable Reads)

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	0xXXXXXXXX																															
Access	RWH																															
Name	QDATA1BIG																															

Bit	Name	Reset	Access	Description
31:0	QDATA1BIG	0xXXXXXXXX	RWH	**Quad Data 1 Big Endian Access**

Big endian access to QDATA1, which is equal to {DATA3, DATA2, DATA1, DATA0} and {DDATA3, DDATA2}. 16x32bits read/write accesses are required to fully read/write QDATA1.

33.6.44 CRYPTO_QDATA0BYTE - QDATA0 Register Byte Access (No Bit Access) (Actionable Reads)

Offset	Bit Position																															
0x1C0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0xXX							
Access																									RWH							
Name																									QDATA0BYTE							

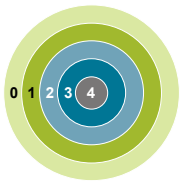
Bit	Name	Reset	Access	Description
31:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7:0	QDATA0BYTE	0xXX	RWH	**Qdata 0 Byte Access**
		Access to QDATA0. 64x8bits read/write accesses are required to fully read/write QDATA0. Accesses must be performed in multiples of 4, or data incoherency may occur		

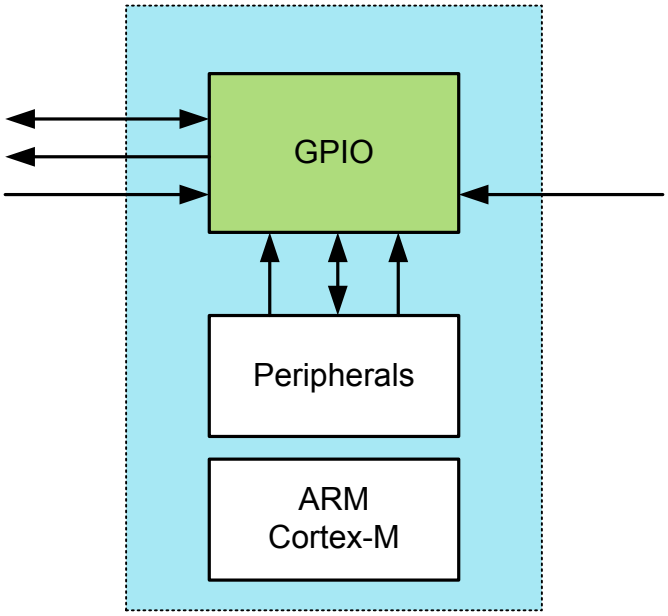
33.6.45 CRYPTO_QDATA1BYTE - QDATA1 Register Byte Access (No Bit Access) (Actionable Reads)

Offset	Bit Position																															
0x1C4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0xXX							
Access																									RWH							
Name																									QDATA1BYTE							

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	QDATA1BYTE	0xXX	RWH	**Qdata 1 Byte Access**
		Access to QDATA1. 64x8bits read/write accesses are required to fully read/write QDATA1. Accesses must be performed in multiples of 4, or data incoherency may occur		

34. 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 several different locations, thus solving congestion issues that may arise with multiple functions on the same pin. Fully asynchronous interrupts can also be generated from any pin.

34.1 Introduction

In the EFM32 Giant Gecko 12 devices the General Purpose Input/Output (GPIO) pins are organized into ports with up to 16 pins each. These GPIO pins can individually be 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. The GPIO pins can also be overridden by peripheral pin connections, like Timer PWM outputs or USART communication, which can be routed to several locations on the device. The GPIO supports up to 16 asynchronous external pin interrupts, which enable interrupts from any pin on the device. Also, the input value of a pin can be routed through the Peripheral Reflex System to other peripherals.

Note: To use the GPIO, the GPIO clock must first be enabled in CMU_HFBUSCLKEN0. Setting this bit enables the HFBUSCLK for the GPIO.

34.2 Features

- Individual configuration for each pin
  - Tristate (reset state)
  - Push-pull
  - Open-drain
  - Pull-up resistor
  - Pull-down resistor
  - Drive strength
    - 1 mA
    - 10 mA
  - Slewrate
  - Over Voltage Tolerance
- EM4 IO pin retention
  - Output enable
  - Output value
  - Pull enable
  - Pull direction
  - Over Voltage Tolerance
- EM4 wake-up on selected GPIO pins
- Glitch suppression input filter
- Alternate functions (e.g. peripheral outputs and inputs)
  - Routed to several locations on the device
  - Pin connections can be enabled individually
  - Output data can be overridden by peripheral
  - Output enable can be overridden by peripheral
- 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 EM0 Active-EM3 Stop
- Peripheral Reflex System producer
  - All GPIO pins are selectable
- Configuration lock functionality to avoid accidental changes

34.3 Functional Description

An overview of the GPIO module is shown in Figure 34.1 Pin Configuration on page 1246. 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 bits 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 34.3.1 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 BITSET address of the GPIO_Px_DOUT register sets the DOUT bits
- Writing the BITCLEAR 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.

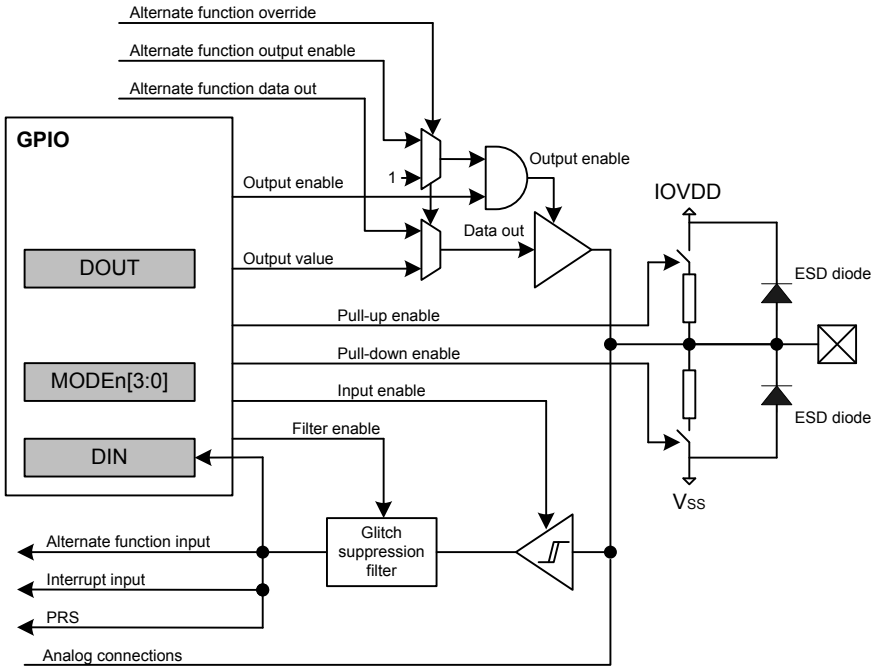


Figure 34.1. Pin Configuration

Note: There is no ESD diode to IOVDD because if using LCD Voltage Boost the pin voltage may be higher than IOVDD. Nevertheless there is an ESD protection block against over voltage.

34.3.1 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 34.1 Pin Configuration on page 1247 shows the available configurations.

Table 34.1. 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, input driver and Over Voltage Tolerance is 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 34.2 Tristated Output With Optional Pull-up or Pull-down on page 1248.

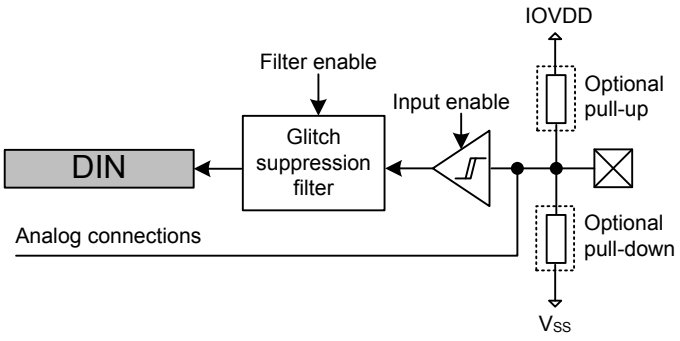


Figure 34.2. Tristated Output With Optional Pull-up or Pull-down

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 34.3 Push-Pull Configuration on page 1248.

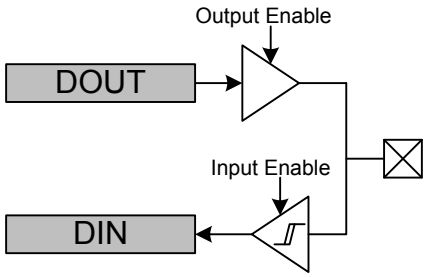


Figure 34.3. Push-Pull Configuration

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 34.4 Open-drain on page 1248. 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.

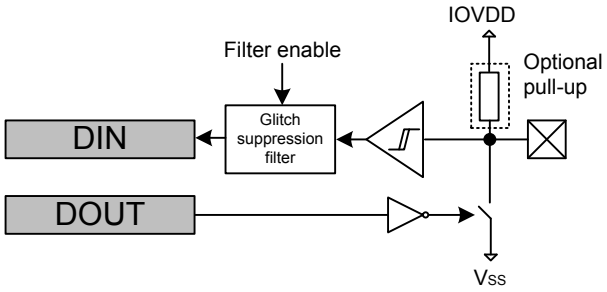


Figure 34.4. Open-drain

34.3.1.1 Over Voltage Tolerance

Over voltage capability is available for most pins. If available, it allows the pin to be used at either the minimum of IOVDD + 2V and 5.5V (for 5V tolerant pads) or the minimum of IOVDD + 2V and 3.8V (for non-5V tolerant pads supporting LCD). The data sheet specifies which pins can be used as 5V tolerant pins. Default over voltage is enabled for each pin supporting that feature. Over voltage tolerance (OVT) can be disabled on a per pin basis. The over voltage tolerance feature applied to the selected pins is configured in the GPIO_Px_OVTDIS register. Disabling the over voltage tolerance for a pin will provide less distortion on that pin, which is useful when the pin is used as analog input.

Note: The VDAC (and OPAMPs) and LCD can drive outputs above IOVDD and therefore the involved pads typically require OVT to be be enabled.

34.3.1.2 Alternate Port Control

The Alternate Port Control allows for additional flexibilty 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 low drive strength setting (normal mode) while the other half uses high drive strength (alternate mode).

Alternate port control is enabled when MODEn is set to any of the ALT enumerated modes (ie. PUSHPULLALT). When MODEn is an alternate mode, the pin uses the alternate port control values specified in the DINDISALT,SLEWRATEALT, and DRIVESTRENGTHALT fields in GPIO_Px_CTRL. In all other modes, the port control values are used from the DINDIS,SLEWRATE, and DRIVESTRENGTH fields in GPIO_Px_CTRL.

34.3.1.3 Drive Strength

The drive strength can be applied to pins on a port-by-port basis. The drive strength applied to pins configured using normal MODEn settings can be controlled using the DRIVESTRENGTH field in GPIO_Px_CTRL. The drive strength applied to pins configured using alternate MODEn settings can be controlled using the DRIVESTRENGTHALT field.

34.3.1.4 Slewrate

The slewrate can be applied to pins on a port-by-port basis. The slewrate 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.

34.3.1.5 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.

34.3.1.6 Configuration Lock

GPIO_Px_MODEL, GPIO_Px_MODEH, GPIO_Px_CTRL, GPIO_Px_PINLOCKN, GPIO_Px_OVTDIS, GPIO_EXTIPSELL, GPIO_EXTIPSELH, GPIO_EXTIPINSELL, GPIO_EXTIPINSELH, GPIO_INSENSE, GPIO_ROUTEPEN, and GPIO_ROUTELOC0 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.

In addition to configuration lock, GPIO_Px_MODEL, GPIO_Px_MODEH, GPIO_Px_DOUT, GPIO_Px_DOUTTGL, and GPIO_Px_OVTDIS can be locked individually for each pin by clearing the corresponding bit in GPIO_Px_PINLOCKN. When a bit in the GPIO_Px_PINLOCKN register is cleared, it will stay cleared until reset.

34.3.2 EM4 Wake-up

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.

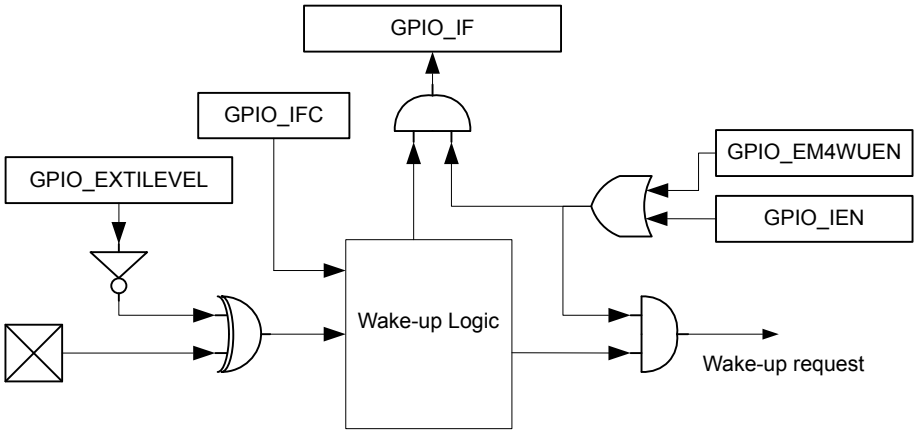


Figure 34.5. EM4 Wake-up Logic

The pins used for EM4 wake-up must be configured as inputs with glitch filters using the GPIO_Px_MODEL/GPIO_Px_MODEH 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. The mapping between EM4WU pins and the bit indexes in the GPIO_EM4WUEN, GPIO_EXTILEVEL, GPIO_IFC, GPIO_IFS, GPIO_IEN, and GPIO_IF registers is as follows:

Table 34.2. EM4WU Register Bit Index to EM4WU Pin Mapping

EM4WU Register Bit Indexes	EM4WU Pin
16	GPIO_EM4WU0
17	GPIO_EM4WU1
18	GPIO_EM4WU2
19	GPIO_EM4WU3
...	...
31	GPIO_EM4WU15
Note: See the device data sheet for actual pin location	

34.3.3 EM4 Retention

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, pull direction and over voltage tolerance while in EM4.

EM4 GPIO retention is controlled with the EM4IORETMODE field in the EMU_EM4CTRL register. Setting EM4IORETMODE to EM4EXIT will cause retention to persist while in EM4 and reset the GPIOs during wakeup. Setting EM4IORETMODE to SWUNLATCH will cause the retention to persist until the EM4UNLATCH bit is written by software. Note that when using SWUNLATCH, the GPIO register values are still reset on wakeup from EM4. In order to ensure that the GPIO state does not change, sofware must re-write the GPIO registers before setting EM4UNLATCH and ending EM4 GPIO retention. See the EMU chapter for additional documentation on its registers and the EM4UNLATCH bit.

34.3.4 Alternate Functions

Alternate functions are connections to pins from peripherals, i.e. Timers, USARTs, etc.. These peripherals contain route registers, where the pin connections are enabled. In addition, the route registers contain a location bit field that configures which pin an output of that peripheral will be connected to if enabled. After connecting a peripheral, the pin configuration stays as set in GPIO_Px_MODEL, GPIO_Px_MODEH and GPIO_Px_DOUT registers. For example, the pin configuration must be set to output enable in GPIO_Px_MODEL or GPIO_Px_MODEH for a peripheral to be able to use the pin as an output.

It is not recommended to select two or more peripherals as output on the same pin. The reader is referred to the pin map section of the device data sheet for more information on the possible locations of each alternate function.

Note:

- Some of the alternate function locations have non-interference priority. These locations prevent the use of the selected pin for other alternate functions. For example, these can be used to secure TIMER PWM outputs from software errors (i.e. another alternate function enabled to the same pin inadvertently).
- Certain alternate functions have high speed priority locations. These locations ensure fastest possible paths to the pins which is useful for timing critical alternate functions. For the alternate function output signals which are using these locations the MODEn must be configured as PUSHPULL or PUSHPULLALT.
- An overview of these locations is provided in the pin map section of the device data sheet.

34.3.4.1 Analog Connections

When using the GPIO pin for analog functionality, it is recommended to disable the over voltage tolerance by setting the corresponding pin in the GPIO_Px_OVTDIS register and setting the MODEn in GPIO_Px_MODEL or GPIO_Px_MODEH equal to DISABLE to disable the input sense, output driver and pull resistors.

34.3.4.2 Debug Connections

34.3.4.2.1 Serial Wire Debug Connection

The SW Debug Port is routed as an alternate function and 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_ROUTEPEN to 0.

34.3.4.2.2 JTAG Debug Connection

The JTAG Debug Port is routed as an alternate function and the TMS, TCK, TDO, and TDI pin connections are enabled by default 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_ROUTEPEN to 0.

34.3.4.2.3 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_ROUTEPEN register can only be updated when the debugger is disconnected from the system. Any attempts to modify GPIO_ROUTEPEN 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 the device after a reset and before the pins are disabled.

34.3.4.2.4 ETM Trace Connections

There are five trace pins available on the device. One trace clock which can be enabled by setting the ETMTCLKPEN bitfield in GPIO_ROUTEPEN. The four data pins can be enabled individually by setting ETMTD0PEN, ETMTD1PEN, ETMTD2PEN and ETMTD3PEN respectively in GPIO_ROUTEPEN. It is possible to choose which pins the trace data will be exported to. The lowest trace bit will be routed to the first enabled trace pin. For example, if the ETM data port size is 2 bits and TD0 and TD3 are enabled, will make bit 0 be routed to TD0 while bit 1 will be routed to TD3.

Both the TCLK and all the TD pins can also be routed to alternate locations by configuring the ETMLOC bitfield in GPIO_ROUTELOC0.

34.3.5 Interrupt Generation

Interrupts may be triggered on edge events for any GPIO pin, or on pin input levels for GPIO capable of EM4 wake-up.

34.3.5.1 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 Stop, see Figure 34.6 Pin N Interrupt Generation on page 1252.

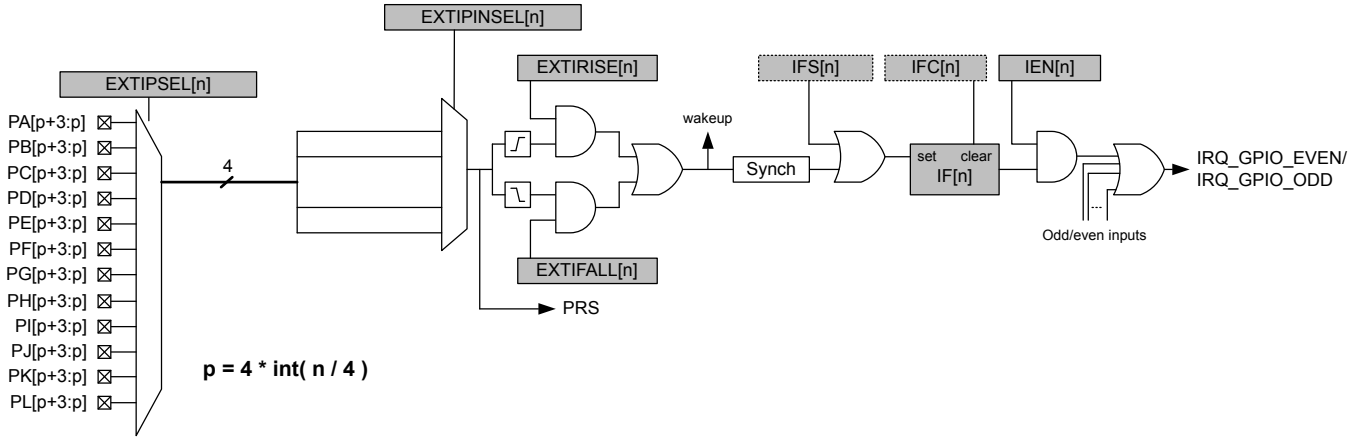


Figure 34.6. Pin N Interrupt Generation

External pin interrupts can be represented in the form of EXTI[index], where index is the external interrupt number. For example, the EXTI7 interrupt has an index of 7. All pins within a group of four (0-3,4-7,8-11,12-15) from all ports are grouped together to trigger one interrupt. The group of pins available to trigger an interrupt is determined by the interrupt index and calculated as int(index/4). For example the first 4 interrupts (EXTI0 - EXTI3) are triggered by pins in the first group (Px[3:0]) and the second 4 interrupts (EXTI4-EXTI7) are triggered by pins in the second group (Px[7:4]).

The EXTIPSELn bits in GPIO_EXTIPSELL or GPIO_EXTIPSELH select which PORT in the group will trigger the interrupt. The EXTIPINSELn bits in GPIO_EXTIPINSELL or GPIO_EXTIPINSELH will determine which pin inside the selected group will trigger the interrupt.

For example if EXTIPSEL11 = PORTB and EXTPINSEL11 = 0 then PB8 will be used for EXTI11. EXTI11 uses the third group (11/4 = 2) so the list of possible pins is Px[11:8]. The setting of EXTIPSEL11 further narrows the selection to PB[11:8]. Finally EXTPINSEL11 selects the first pin in that group which is PB8.

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 MODEL and MODEH fields in the GPIO_Px_MODEL and GPIO_Px_MODEH registers, respectively, should be set to include glitch filtering for pins that have external interrupts enabled.

34.3.5.2 Level Interrupt Generation

GPIO can generate a level interrupt using the input of any GPIO EM4 wake-up pins on the device. The interrupts have asynchronous sense capability, enabling wake-up from energy modes as low as EM4.

In order to enable the level interrupt, set the EM4WU field in the GPIO_IEN register and the EM4WUn field in the GPIO_EXTILEVEL register. Upon a level interrupt occuring, 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 granulalrity 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 34.3 Level Interrupt Energy Mode Wakeup on page 1253

Table 34.3. Level Interrupt Energy Mode Wakeup

GPIO_IEN	GPIO_EM4WUEN	Energy Mode Wakeup
0	0	No Interrupt
0	1	EM4H,EM4S
1	0	EM1,EM2,EM3,EM4H,EM4S
1	1	EM1,EM2,EM3,EM4H,EM4S

34.3.6 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 Figure 34.6 Pin N Interrupt Generation on page 1252 for an illustration of where the PRS output signal is generated.

34.3.7 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 the HFBUSCLK. Consequently, when a pin changes state, the change will have propagated 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 INT or EM4WU,respectively, in GPIO_INSENSE register.

34.4 Register Map

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

Offset	Name	Type	Description
0x000	GPIO_PA_CTRL	RW	Port Control Register
0x004	GPIO_PA_MODEL	RW	Port Pin Mode Low Register
0x008	GPIO_PA_MODEH	RW	Port Pin Mode High Register
0x00C	GPIO_PA_DOUT	RW	Port Data Out Register
0x018	GPIO_PA_DOUTTGL	W1	Port Data Out Toggle Register
0x01C	GPIO_PA_DIN	R	Port Data in Register
0x020	GPIO_PA_PINLOCKN	RW	Port Unlocked Pins Register
0x028	GPIO_PA_OVTDIS	RW	Over Voltage Disable for All Modes
...	GPIO_Px_CTRL	RW	Port Control Register
...	GPIO_Px_MODEL	RW	Port Pin Mode Low Register
...	GPIO_Px_MODEH	RW	Port Pin Mode High Register
...	GPIO_Px_DOUT	RW	Port Data Out Register
...	GPIO_Px_DOUTTGL	W1	Port Data Out Toggle Register
...	GPIO_Px_DIN	R	Port Data in Register
...	GPIO_Px_PINLOCKN	RW	Port Unlocked Pins Register
...	GPIO_Px_OVTDIS	RW	Over Voltage Disable for All Modes
0x210	GPIO_PL_CTRL	RW	Port Control Register
0x214	GPIO_PL_MODEL	RW	Port Pin Mode Low Register
0x218	GPIO_PL_MODEH	RW	Port Pin Mode High Register
0x21C	GPIO_PL_DOUT	RW	Port Data Out Register
0x228	GPIO_PL_DOUTTGL	W1	Port Data Out Toggle Register
0x22C	GPIO_PL_DIN	R	Port Data in Register
0x230	GPIO_PL_PINLOCKN	RW	Port Unlocked Pins Register
0x238	GPIO_PL_OVTDIS	RW	Over Voltage Disable for All Modes
0x400	GPIO_EXTIPSELL	RW	External Interrupt Port Select Low Register
0x404	GPIO_EXTIPSELH	RW	External Interrupt Port Select High Register
0x408	GPIO_EXTIPINSELL	RW	External Interrupt Pin Select Low Register
0x40C	GPIO_EXTIPINSELH	RW	External Interrupt Pin Select High Register
0x410	GPIO_EXTIRISE	RW	External Interrupt Rising Edge Trigger Register
0x414	GPIO_EXTIFALL	RW	External Interrupt Falling Edge Trigger Register
0x418	GPIO_EXTILEVEL	RW	External Interrupt Level Register
0x41C	GPIO_IF	R	Interrupt Flag Register
0x420	GPIO_IFS	W1	Interrupt Flag Set Register
0x424	GPIO_IFC	(R)W1	Interrupt Flag Clear Register
0x428	GPIO_IEN	RW	Interrupt Enable Register

Offset	Name	Type	Description
0x42C	GPIO_EM4WUEN	RW	EM4 Wake Up Enable Register
0x440	GPIO_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x444	GPIO_ROUTELOC0	RW	I/O Routing Location Register
0x450	GPIO_INSENSE	RW	Input Sense Register
0x454	GPIO_LOCK	RWH	Configuration Lock Register

34.5 Register Description

34.5.1 GPIO_Px_CTRL - Port 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				0							0x5					0				0							0x5					0
Access				RW							RW					RW				RW							RW					RW
Name				DINDISALT							SLEWRATEALT					DRIVESTRENGTHALT				DINDIS							SLEWRATE					DRIVESTRENGTH

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	0	RW	**Alternate Data in Disable**
	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	0x5	RW	**Alternate Slewrate Limit for Port**
	Slewrate limit for port pins using alternate modes. Higher values represent faster slewrates.			
19:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	DRIVESTRENGTHALT	0	RW	**Alternate Drive Strength for Port**
	Drive strength setting for port pins using alternate drive strength.			

Value	Mode	Description
0	STRONG	10 mA drive current
1	WEAK	1 mA drive current

15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	DINDIS	0	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	0x5	RW	**Slewrate Limit for Port**
	Slewrate limit for port pins not using alternate modes. Higher values represent faster slewrates.			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
0	DRIVESTRENGTH	0	RW	**Drive Strength for Port**

Drive strength setting for port pins not using alternate modes.

Value	Mode	Description
0	STRONG	10 mA drive current
1	WEAK	1 mA drive current

34.5.2 GPIO_Px_MODEL - Port Pin Mode Low 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			
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	**Pin 7 Mode**

Configure mode for pin 7.

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	WIREDANDPULLUP-FILTER	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	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output using alternate control with filter and pullup

Bit	Name	Reset	Access	Description
27:24	MODE6	0x0	RW	**Pin 6 Mode**

Configure mode for pin 6.

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	WIREDANDPULLUP-FILTER		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	WIREDANDALTPULL-UP		Open-drain output using alternate control with pullup
	15	WIREDANDALTPUL-LUPFILTER		Open-drain output using alternate control with filter and pullup
23:20	MODE5	0x0	RW	**Pin 5 Mode**
	Configure mode for pin 5.			

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	WIREDANDPULLUP-FILTER	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	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output using alternate control with filter and pullup

Bit	Name	Reset	Access	Description
19:16	MODE4	0x0	RW	**Pin 4 Mode**

Configure mode for pin 4.

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	WIREDANDPULLUP-FILTER	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	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output using alternate control with filter and pullup

15:12	MODE3	0x0	RW	**Pin 3 Mode**

Configure mode for pin 3.

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	WIREDANDPULLUP-FILTER	Open-drain output with filter and pullup

Bit	Name	Reset	Access	Description
	12	WIREDANDALT		Open-drain output using alternate control
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter
	14	WIREDANDALTPULL-UP		Open-drain output using alternate control with pullup
	15	WIREDANDALTPUL-LUPFILTER		Open-drain output using alternate control with filter and pullup
11:8	MODE2	0x0	RW	**Pin 2 Mode**
	Configure mode for pin 2.			
	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	WIREDANDPULLUP-FILTER		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	WIREDANDALTPULL-UP		Open-drain output using alternate control with pullup
	15	WIREDANDALTPUL-LUPFILTER		Open-drain output using alternate control with filter and pullup
7:4	MODE1	0x0	RW	**Pin 1 Mode**
	Configure mode for pin 1.			
	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

Bit	Name	Reset	Access	Description
	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	WIREDANDPULLUP-FILTER		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	WIREDANDALTPULL-UP		Open-drain output using alternate control with pullup
	15	WIREDANDALTPUL-LUPFILTER		Open-drain output using alternate control with filter and pullup
3:0	MODE0	0x0	RW	**Pin 0 Mode**

Configure mode for pin 0.

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	WIREDANDPULLUP-FILTER	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	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output using alternate control with filter and pullup

34.5.3 GPIO_Px_MODEH - Port Pin Mode High 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			
Access	RW				RW				RW				RW				RW				RW				RW				RW			
Name	MODE15				MODE14				MODE13				MODE12				MODE11				MODE10				MODE9				MODE8			

Bit	Name	Reset	Access	Description
31:28	MODE15	0x0	RW	**Pin 15 Mode**

Configure mode for pin 15.

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	WIREDANDPULLUP-FILTER	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	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output using alternate control with filter and pullup

Bit	Name	Reset	Access	Description
27:24	MODE14	0x0	RW	**Pin 14 Mode**

Configure mode for pin 14.

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	WIREDANDPULLUP-FILTER		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	WIREDANDALTPULL-UP		Open-drain output using alternate control with pullup
	15	WIREDANDALTPUL-LUPFILTER		Open-drain output using alternate control with filter and pullup
23:20	MODE13	0x0	RW	**Pin 13 Mode**
	Configure mode for pin 13.			

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	WIREDANDPULLUP-FILTER	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	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output using alternate control with filter and pullup

Bit	Name	Reset	Access	Description
19:16	MODE12	0x0	RW	**Pin 12 Mode**

Configure mode for pin 12.

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	WIREDANDPULLUP-FILTER	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	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output using alternate control with filter and pullup

15:12	MODE11	0x0	RW	**Pin 11 Mode**

Configure mode for pin 11.

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	WIREDANDPULLUP-FILTER	Open-drain output with filter and pullup

Bit	Name	Reset	Access	Description
	12	WIREDANDALT		Open-drain output using alternate control
	13	WIREDANDALTFILTER		Open-drain output using alternate control with filter
	14	WIREDANDALTPULL-UP		Open-drain output using alternate control with pullup
	15	WIREDANDALTPUL-LUPFILTER		Open-drain output using alternate control with filter and pullup
11:8	MODE10	0x0	RW	**Pin 10 Mode**
	Configure mode for pin 10.			
	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	WIREDANDPULLUP-FILTER		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	WIREDANDALTPULL-UP		Open-drain output using alternate control with pullup
	15	WIREDANDALTPUL-LUPFILTER		Open-drain output using alternate control with filter and pullup
7:4	MODE9	0x0	RW	**Pin 9 Mode**
	Configure mode for pin 9.			
	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

Bit	Name	Reset	Access	Description
	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	WIREDANDPULLUP-FILTER		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	WIREDANDALTPULL-UP		Open-drain output using alternate control with pullup
	15	WIREDANDALTPUL-LUPFILTER		Open-drain output using alternate control with filter and pullup
3:0	MODE8	0x0	RW	**Pin 8 Mode**

Configure mode for pin 8.

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	WIREDANDPULLUP-FILTER	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	WIREDANDALTPULL-UP	Open-drain output using alternate control with pullup
15	WIREDANDALTPUL-LUPFILTER	Open-drain output using alternate control with filter and pullup

34.5.4 GPIO_Px_DOUT - Port Data Out 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																	0x0000															
Access																	RW															
Name																	DOUT															

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	DOUT	0x0000	RW	**Data Out**
	Data output on pin.			

34.5.5 GPIO_Px_DOUTTGL - Port Data Out Toggle 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																	0x0000															
Access																	W1															
Name																	DOUTTGL															

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	DOUTTGL	0x0000	W1	**Data Out Toggle**
	Write bits to 1 to toggle corresponding bits in GPIO_Px_DOUT. Bits written to 0 will have no effect.			

34.5.6 GPIO_Px_DIN - Port Data in 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																	0x0000															
Access																	R															
Name																	DIN															

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	DIN	0x0000	R	**Data in**
	Port data input.			

34.5.7 GPIO_Px_PINLOCKN - Port Unlocked Pins 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																	0xFFFF															
Access																	RW															
Name																	PINLOCKN															

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	PINLOCKN	0xFFFF	RW	**Unlocked Pins**
	Shows unlocked pins in the port. To lock pin n, clear bit n. The pin is then locked until reset.			

34.5.8 GPIO_Px_OVTDIS - Over Voltage Disable for All Modes

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																	0x0000															
Access																	RW															
Name																	OVTDIS															

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	OVTDIS	0x0000	RW	**Disable Over Voltage Capability** Disabling the Over Voltage capability will provide less distortion on analog inputs.

34.5.9 GPIO_EXTIPSELL - External Interrupt Port Select Low 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				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
31:28	EXTIPSEL7	0x0	RW	**External Interrupt 7 Port Select**

Select input port for external interrupt 7.

Value	Mode	Description
0	PORTA	Port A group selected for external interrupt 7
1	PORTB	Port B group selected for external interrupt 7
2	PORTC	Port C group selected for external interrupt 7
3	PORTD	Port D group selected for external interrupt 7
4	PORTE	Port E group selected for external interrupt 7
5	PORTF	Port F group selected for external interrupt 7

Bit	Name	Reset	Access	Description
27:24	EXTIPSEL6	0x0	RW	**External Interrupt 6 Port Select**

Select input port for external interrupt 6.

Value	Mode	Description
0	PORTA	Port A group selected for external interrupt 6
1	PORTB	Port B group selected for external interrupt 6
2	PORTC	Port C group selected for external interrupt 6
3	PORTD	Port D group selected for external interrupt 6
4	PORTE	Port E group selected for external interrupt 6
5	PORTF	Port F group selected for external interrupt 6

Bit	Name	Reset	Access	Description
23:20	EXTIPSEL5	0x0	RW	**External Interrupt 5 Port Select**

Select input port for external interrupt 5.

Value	Mode	Description
0	PORTA	Port A group selected for external interrupt 5
1	PORTB	Port B group selected for external interrupt 5
2	PORTC	Port C group selected for external interrupt 5
3	PORTD	Port D group selected for external interrupt 5
4	PORTE	Port E group selected for external interrupt 5

Bit	Name	Reset	Access	Description
	5	PORTF		Port F group selected for external interrupt 5
19:16	EXTIPSEL4	0x0	RW	**External Interrupt 4 Port Select**
	Select input port for external interrupt 4.			
	Value	Mode		Description
	0	PORTA		Port A group selected for external interrupt 4
	1	PORTB		Port B group selected for external interrupt 4
	2	PORTC		Port C group selected for external interrupt 4
	3	PORTD		Port D group selected for external interrupt 4
	4	PORTE		Port E group selected for external interrupt 4
	5	PORTF		Port F group selected for external interrupt 4
15:12	EXTIPSEL3	0x0	RW	**External Interrupt 3 Port Select**
	Select input port for external interrupt 3.			
	Value	Mode		Description
	0	PORTA		Port A group selected for external interrupt 3
	1	PORTB		Port B group selected for external interrupt 3
	2	PORTC		Port C group selected for external interrupt 3
	3	PORTD		Port D group selected for external interrupt 3
	4	PORTE		Port E group selected for external interrupt 3
	5	PORTF		Port F group selected for external interrupt 3
11:8	EXTIPSEL2	0x0	RW	**External Interrupt 2 Port Select**
	Select input port for external interrupt 2.			
	Value	Mode		Description
	0	PORTA		Port A group selected for external interrupt 2
	1	PORTB		Port B group selected for external interrupt 2
	2	PORTC		Port C group selected for external interrupt 2
	3	PORTD		Port D group selected for external interrupt 2
	4	PORTE		Port E group selected for external interrupt 2
	5	PORTF		Port F group selected for external interrupt 2
7:4	EXTIPSEL1	0x0	RW	**External Interrupt 1 Port Select**
	Select input port for external interrupt 1.			
	Value	Mode		Description
	0	PORTA		Port A group selected for external interrupt 1
	1	PORTB		Port B group selected for external interrupt 1
	2	PORTC		Port C group selected for external interrupt 1
	3	PORTD		Port D group selected for external interrupt 1

Bit	Name	Reset	Access	Description
	4	PORTE		Port E group selected for external interrupt 1
	5	PORTF		Port F group selected for external interrupt 1
3:0	EXTIPSEL0	0x0	RW	**External Interrupt 0 Port Select**

Select input port for external interrupt 0.

Value	Mode	Description
0	PORTA	Port A group selected for external interrupt 0
1	PORTB	Port B group selected for external interrupt 0
2	PORTC	Port C group selected for external interrupt 0
3	PORTD	Port D group selected for external interrupt 0
4	PORTE	Port E group selected for external interrupt 0
5	PORTF	Port F group selected for external interrupt 0

34.5.10 GPIO_EXTIPSELH - External Interrupt Port Select High Register

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				0x0				0x0				0x0				0x0			
Access	RW				RW				RW				RW				RW				RW				RW				RW			
Name	EXTIPSEL15				EXTIPSEL14				EXTIPSEL13				EXTIPSEL12				EXTIPSEL11				EXTIPSEL10				EXTIPSEL9				EXTIPSEL8			

Bit	Name	Reset	Access	Description
31:28	EXTIPSEL15	0x0	RW	**External Interrupt 15 Port Select**

Select input port for external interrupt 15.

Value	Mode	Description
0	PORTA	Port A group selected for external interrupt 15
1	PORTB	Port B group selected for external interrupt 15
2	PORTC	Port C group selected for external interrupt 15
3	PORTD	Port D group selected for external interrupt 15
4	PORTE	Port E group selected for external interrupt 15
5	PORTF	Port F group selected for external interrupt 15

Bit	Name	Reset	Access	Description
27:24	EXTIPSEL14	0x0	RW	**External Interrupt 14 Port Select**

Select input port for external interrupt 14.

Value	Mode	Description
0	PORTA	Port A group selected for external interrupt 14
1	PORTB	Port B group selected for external interrupt 14
2	PORTC	Port C group selected for external interrupt 14
3	PORTD	Port D group selected for external interrupt 14
4	PORTE	Port E group selected for external interrupt 14
5	PORTF	Port F group selected for external interrupt 14

Bit	Name	Reset	Access	Description
23:20	EXTIPSEL13	0x0	RW	**External Interrupt 13 Port Select**

Select input port for external interrupt 13.

Value	Mode	Description
0	PORTA	Port A group selected for external interrupt 13
1	PORTB	Port B group selected for external interrupt 13
2	PORTC	Port C group selected for external interrupt 13
3	PORTD	Port D group selected for external interrupt 13
4	PORTE	Port E group selected for external interrupt 13

Bit	Name	Reset	Access	Description
	5	PORTF		Port F group selected for external interrupt 13
19:16	EXTIPSEL12	0x0	RW	**External Interrupt 12 Port Select**
	Select input port for external interrupt 12.			
	Value	Mode		Description
	0	PORTA		Port A group selected for external interrupt 12
	1	PORTB		Port B group selected for external interrupt 12
	2	PORTC		Port C group selected for external interrupt 12
	3	PORTD		Port D group selected for external interrupt 12
	4	PORTE		Port E group selected for external interrupt 12
	5	PORTF		Port F group selected for external interrupt 12
15:12	EXTIPSEL11	0x0	RW	**External Interrupt 11 Port Select**
	Select input port for external interrupt 11.			
	Value	Mode		Description
	0	PORTA		Port A group selected for external interrupt 11
	1	PORTB		Port B group selected for external interrupt 11
	2	PORTC		Port C group selected for external interrupt 11
	3	PORTD		Port D group selected for external interrupt 11
	4	PORTE		Port E group selected for external interrupt 11
	5	PORTF		Port F group selected for external interrupt 11
11:8	EXTIPSEL10	0x0	RW	**External Interrupt 10 Port Select**
	Select input port for external interrupt 10.			
	Value	Mode		Description
	0	PORTA		Port A group selected for external interrupt 10
	1	PORTB		Port B group selected for external interrupt 10
	2	PORTC		Port C group selected for external interrupt 10
	3	PORTD		Port D group selected for external interrupt 10
	4	PORTE		Port E group selected for external interrupt 10
	5	PORTF		Port F group selected for external interrupt 10
7:4	EXTIPSEL9	0x0	RW	**External Interrupt 9 Port Select**
	Select input port for external interrupt 9.			
	Value	Mode		Description
	0	PORTA		Port A group selected for external interrupt 9
	1	PORTB		Port B group selected for external interrupt 9
	2	PORTC		Port C group selected for external interrupt 9
	3	PORTD		Port D group selected for external interrupt 9

Bit	Name	Reset	Access	Description
	4	PORTE		Port E group selected for external interrupt 9
	5	PORTF		Port F group selected for external interrupt 9
3:0	EXTIPSEL8	0x0	RW	**External Interrupt 8 Port Select**

Select input port for external interrupt 8.

Value	Mode	Description
0	PORTA	Port A group selected for external interrupt 8
1	PORTB	Port B group selected for external interrupt 8
2	PORTC	Port C group selected for external interrupt 8
3	PORTD	Port D group selected for external interrupt 8
4	PORTE	Port E group selected for external interrupt 8
5	PORTF	Port F group selected for external interrupt 8

34.5.11 GPIO_EXTIPINSELL - External Interrupt Pin Select Low 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			0x3				0x2				0x1				0x0				0x3				0x2				0x1				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	0x3	RW	**External Interrupt 7 Pin Select**

Select the pin for external interrupt 7.

Value	Mode	Description
0	PIN4	Pin 4
1	PIN5	Pin 5
2	PIN6	Pin 6
3	PIN7	Pin 7

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	0x2	RW	**External Interrupt 6 Pin Select**

Select the pin for external interrupt 6.

Value	Mode	Description
0	PIN4	Pin 4
1	PIN5	Pin 5
2	PIN6	Pin 6
3	PIN7	Pin 7

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	0x1	RW	**External Interrupt 5 Pin Select**

Select the pin for external interrupt 5.

Value	Mode	Description
0	PIN4	Pin 4
1	PIN5	Pin 5
2	PIN6	Pin 6
3	PIN7	Pin 7

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 4 Pin Select**

Select the pin for external interrupt 4.

Value	Mode	Description
0	PIN4	Pin 4
1	PIN5	Pin 5
2	PIN6	Pin 6
3	PIN7	Pin 7

15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	EXTIPINSEL3	0x3	RW	**External Interrupt 3 Pin Select**

Select the pin for external interrupt 3.

Value	Mode	Description
0	PIN0	Pin 0
1	PIN1	Pin 1
2	PIN2	Pin 2
3	PIN3	Pin 3

11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	EXTIPINSEL2	0x2	RW	**External Interrupt 2 Pin Select**

Select the pin for external interrupt 2.

Value	Mode	Description
0	PIN0	Pin 0
1	PIN1	Pin 1
2	PIN2	Pin 2
3	PIN3	Pin 3

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	EXTIPINSEL1	0x1	RW	**External Interrupt 1 Pin Select**

Select the pin for external interrupt 1.

Value	Mode	Description
0	PIN0	Pin 0
1	PIN1	Pin 1
2	PIN2	Pin 2
3	PIN3	Pin 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 0 Pin Select**

Select the pin for external interrupt 0.

Value	Mode	Description
0	PIN0	Pin 0
1	PIN1	Pin 1
2	PIN2	Pin 2
3	PIN3	Pin 3

34.5.12 GPIO_EXTIPINSELH - External Interrupt Pin Select High Register

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			0x3				0x2				0x1				0x0				0x3				0x2				0x1				0x0	
Access			RW				RW				RW				RW				RW				RW				RW				RW	
Name			EXTIPINSEL15				EXTIPINSEL14				EXTIPINSEL13				EXTIPINSEL12				EXTIPINSEL11				EXTIPINSEL10				EXTIPINSEL9				EXTIPINSEL8	

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	EXTIPINSEL15	0x3	RW	**External Interrupt 15 Pin Select**

Select the pin for external interrupt 15.

Value	Mode	Description
0	PIN12	Pin 12
1	PIN13	Pin 13
2	PIN14	Pin 14
3	PIN15	Pin 15

27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:24	EXTIPINSEL14	0x2	RW	**External Interrupt 14 Pin Select**

Select the pin for external interrupt 14.

Value	Mode	Description
0	PIN12	Pin 12
1	PIN13	Pin 13
2	PIN14	Pin 14
3	PIN15	Pin 15

23:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	EXTIPINSEL13	0x1	RW	**External Interrupt 13 Pin Select**

Select the pin for external interrupt 13.

Value	Mode	Description
0	PIN12	Pin 12
1	PIN13	Pin 13
2	PIN14	Pin 14
3	PIN15	Pin 15

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	EXTIPINSEL12	0x0	RW	**External Interrupt 12 Pin Select**

Select the pin for external interrupt 12.

Value	Mode	Description
0	PIN12	Pin 12
1	PIN13	Pin 13
2	PIN14	Pin 14
3	PIN15	Pin 15

15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	EXTIPINSEL11	0x3	RW	**External Interrupt 11 Pin Select**

Select the pin for external interrupt 11.

Value	Mode	Description
0	PIN8	Pin 8
1	PIN9	Pin 9
2	PIN10	Pin 10
3	PIN11	Pin 11

11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:8	EXTIPINSEL10	0x2	RW	**External Interrupt 10 Pin Select**

Select the pin for external interrupt 10.

Value	Mode	Description
0	PIN8	Pin 8
1	PIN9	Pin 9
2	PIN10	Pin 10
3	PIN11	Pin 11

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	EXTIPINSEL9	0x1	RW	**External Interrupt 9 Pin Select**

Select the pin for external interrupt 9.

Value	Mode	Description
0	PIN8	Pin 8
1	PIN9	Pin 9
2	PIN10	Pin 10
3	PIN11	Pin 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	EXTIPINSEL8	0x0	RW	**External Interrupt 8 Pin Select**

Select the pin for external interrupt 8.

Value	Mode	Description
0	PIN8	Pin 8
1	PIN9	Pin 9
2	PIN10	Pin 10
3	PIN11	Pin 11

34.5.13 GPIO_EXTIRISE - External Interrupt Rising Edge Trigger Register

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																	0x0000															
Access																	RW															
Name																	EXTIRISE															

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	EXTIRISE	0x0000	RW	**External Interrupt N Rising Edge Trigger Enable**

Set bit n to enable triggering of external interrupt n on rising edge.

Value	Description
EXTIRISE[n] = 0	Rising edge trigger disabled
EXTIRISE[n] = 1	Rising edge trigger enabled

34.5.14 GPIO_EXTIFALL - External Interrupt Falling Edge Trigger Register

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																	0x0000															
Access																	RW															
Name																	EXTIFALL															

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	EXTIFALL	0x0000	RW	**External Interrupt N Falling Edge Trigger Enable**

Set bit n to enable triggering of external interrupt n on falling edge.

Value	Description
EXTIFALL[n] = 0	Falling edge trigger disabled
EXTIFALL[n] = 1	Falling edge trigger enabled

34.5.15 GPIO_EXTILEVEL - External Interrupt Level Register

Offset	Bit Position																													
0x418	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset							0	0	0	0	0	0	0	0	0	0																
Access							RW	RW	RW	RW	RW	RW	RW	RW	RW	RW																
Name							EM4WU9	EM4WU8	EM4WU7	EM4WU6	EM4WU5	EM4WU4	EM4WU3	EM4WU2	EM4WU1	EM4WU0																

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	EM4WU9	0	RW	**EM4 Wake Up Level for EM4WU9 Pin**
24	EM4WU8	0	RW	**EM4 Wake Up Level for EM4WU8 Pin**
23	EM4WU7	0	RW	**EM4 Wake Up Level for EM4WU7 Pin**
22	EM4WU6	0	RW	**EM4 Wake Up Level for EM4WU6 Pin**
21	EM4WU5	0	RW	**EM4 Wake Up Level for EM4WU5 Pin**
20	EM4WU4	0	RW	**EM4 Wake Up Level for EM4WU4 Pin**
19	EM4WU3	0	RW	**EM4 Wake Up Level for EM4WU3 Pin**
18	EM4WU2	0	RW	**EM4 Wake Up Level for EM4WU2 Pin**
17	EM4WU1	0	RW	**EM4 Wake Up Level for EM4WU1 Pin**
16	EM4WU0	0	RW	**EM4 Wake Up Level for EM4WU0 Pin**
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

34.5.16 GPIO_IF - Interrupt Flag Register

Offset	Bit Position																													
0x41C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0000																0x0000															
Access	R																R															
Name	EM4WU																EXT															

Bit	Name	Reset	Access	Description
31:16	EM4WU	0x0000	R	**EM4 Wake Up Pin Interrupt Flag**
	EM4 wake up Pin Interrupt flag.			

Value	Description
0	Interrupt flag cleared
1	Interrupt flag set

Bit	Name	Reset	Access	Description
15:0	EXT	0x0000	R	**External Pin Interrupt Flag**
	Pin n external interrupt flag.			

Value	Description
0	External interrupt flag cleared
1	External interrupt flag set

34.5.17 GPIO_IFS - Interrupt Flag Set Register

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	0x0000																0x0000															
Access	W1																W1															
Name	EM4WU																EXT															

Bit	Name	Reset	Access	Description
31:16	EM4WU	0x0000	W1	**Set EM4WU Interrupt Flag**
	Write 1 to set the EM4WU interrupt flag			
15:0	EXT	0x0000	W1	**Set EXT Interrupt Flag**
	Write 1 to set the EXT interrupt flag			

34.5.18 GPIO_IFC - Interrupt Flag Clear Register

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	0x0000																0x0000															
Access	(R)W1																(R)W1															
Name	EM4WU																EXT															

Bit	Name	Reset	Access	Description
31:16	EM4WU	0x0000	(R)W1	**Clear EM4WU Interrupt Flag**
	Write 1 to clear the EM4WU interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
15:0	EXT	0x0000	(R)W1	**Clear EXT Interrupt Flag**
	Write 1 to clear the EXT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

34.5.19 GPIO_IEN - Interrupt Enable Register

Offset	Bit Position																													
0x428	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0000																0x0000															
Access	RW																RW															
Name	EM4WU																EXT															

Bit	Name	Reset	Access	Description
31:16	EM4WU	0x0000	RW	**EM4WU Interrupt Enable**
	Enable/disable the EM4WU interrupt			
15:0	EXT	0x0000	RW	**EXT Interrupt Enable**
	Enable/disable the EXT interrupt			

34.5.20 GPIO_EM4WUEN - EM4 Wake Up Enable Register

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	0x0000																															
Access	RW																															
Name	EM4WUEN																															

Bit	Name	Reset	Access	Description
31:16	EM4WUEN	0x0000	RW	**EM4 Wake Up Enable**

Write 1 to enable EM4 wake up request, write 0 to disable EM4 wake up request.

Value	Description
0	Disable EM4 wake up on pin
1	Enable EM4 wake up on pin

15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*

34.5.21 GPIO_ROUTEPEN - I/O Routing Pin Enable Register

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												0	0	0	0	0												0	1	1	1	1
Access												RW	RW	RW	RW	RW												RW	RW	RW	RW	RW
Name												ETMTD3PEN	ETMTD2PEN	ETMTD1PEN	ETMTD0PEN	ETMTCLKPEN												SWVPEN	TDIPEN	TDOPEN	SWDIOTMSPEN	SWCLKTCKPEN

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	ETMTD3PEN	0	RW	**ETM Trace Data Pin Enable**
	Enable ETM Trace Data Output 3 connection to pin.			
19	ETMTD2PEN	0	RW	**ETM Trace Data Pin Enable**
	Enable ETM Trace Data Output 2 connection to pin.			
18	ETMTD1PEN	0	RW	**ETM Trace Data Pin Enable**
	Enable ETM Trace Data Output 1 connection to pin.			
17	ETMTD0PEN	0	RW	**ETM Trace Data Pin Enable**
	Enable ETM Trace Data Output 0 connection to pin.			
16	ETMTCLKPEN	0	RW	**ETM Trace Clock Pin Enable**
	Enable ETM Trace Clock Output connection to pin.			
15:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	SWVPEN	0	RW	**Serial Wire Viewer Output Pin Enable**
	Enable Serial Wire Viewer connection to pin.			
3	TDIPEN	1	RW	**JTAG Test Debug Input Pin Enable**
	Enable JTAG TDI connection to pin.			
2	TDOPEN	1	RW	**JTAG Test Debug Output Pin Enable**
	Enable JTAG TDO connection to pin.			
1	SWDIOTMSPEN	1	RW	**Serial Wire Data and JTAG Test Mode Select Pin Enable**
	Enable Serial Wire Data and JTAG Test Mode Select connection to pin. WARNING: When this 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 you 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	1	RW	**Serial Wire Clock and JTAG Test Clock Pin Enable**
	Enable Serial Wire and JTAG Clock connection to pin. WARNING: When this 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 you 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.			

34.5.22 GPIO_ROUTELOC0 - I/O Routing Location Register

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																					0x00						0x00					
Access																					RW						RW					
Name																					ETMLOC						SWVLOC					

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*		
11:6	ETMLOC	0x00	RW	**I/O Location**

Decides the location of the ETM pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4

Bit	Name	Reset	Access	Description
5:0	SWVLOC	0x00	RW	**I/O Location**

Decides the location of the SWV pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

34.5.23 GPIO_INSENSE - Input Sense Register

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																															1	1
Access																															RW	RW
Name																															EM4WU	INT

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	EM4WU	1	RW	**EM4WU Interrupt Sense Enable**
	Set this bit to enable input sensing for EM4WU interrupts.			
0	INT	1	RW	**Interrupt Sense Enable**
	Set this bit to enable input sensing for interrupts.			

34.5.24 GPIO_LOCK - Configuration Lock Register

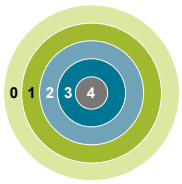
Offset	Bit Position																													
0x454	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0000															
Access																	RWH															
Name																	LOCKKEY															

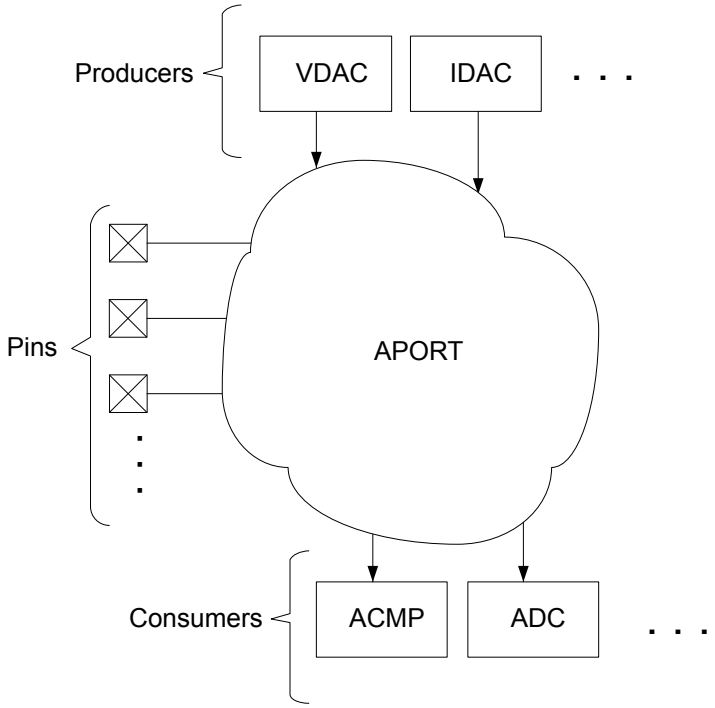
Bit	Name	Reset	Access	Description
31:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:0	LOCKKEY	0x0000	RWH	**Configuration Lock Key**

Write any other value than the unlock code to lock MODEL, MODEH, CTRL, PINLOCKN, OVTDIS, EXTIPSELL, EXTIPSELH, EXTIGSELL, EXTIGSELH,INSENSE, ROUTEPEN, and ROUTELOC0 from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.

Mode	Value	Description
Read Operation		
UNLOCKED	0	GPIO registers are unlocked
LOCKED	1	GPIO registers are locked
Write Operation		
LOCK	0	Lock GPIO registers
UNLOCK	0xA534	Unlock GPIO registers

35. APORT - Analog Port





Quick Facts

What?

The Analog Port (APORT) is a set of analog buses which are used to connect I/O pins to analog peripheral signals.

Why?

The APORT gives on-chip analog resources access to a large number of I/O pins, and provides the system designer with a high degree of routing flexibility.

How?

An analog peripheral requests a pad by simply configuring its input/output to use a channel on APORT. That selection becomes an APORT request where the APORT control switches the pad and the analog signal onto a common bus.

35.1 Introduction

APORT consists of wires, switches, and control logic needed to route signals between analog peripherals and I/O pins. On-chip clients can be either producers or consumers. Analog producers are active devices that drive current/voltage into an APORT, such as current or voltage DACs. Consumers are passive devices that monitor or react to the current/voltage routed to them via the APORT, such as ADCs or analog comparators (ACMP).

35.2 Features

- Pins are typically mapped to two different APORT buses
- Arbitration and conflict status provided to each APORT client

35.3 Functional Description

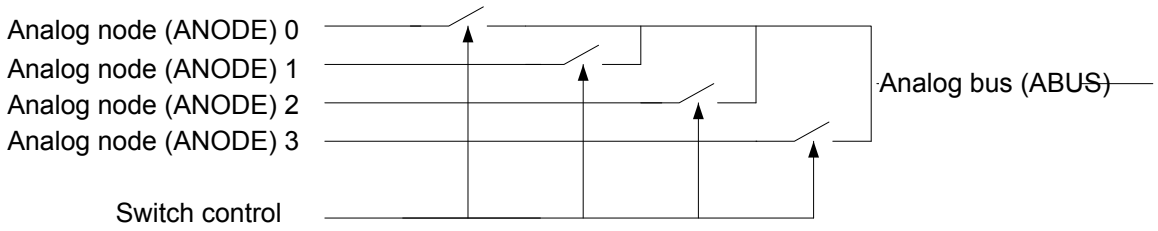


Figure 35.1. Analog Bus (ABUS)

An analog bus (ABUS) consists of analog switches connected to a common wire as shown in Figure 35.1 Analog Bus (ABUS) on page 1293. An APORT consists of multiple ABUSes. Since many clients can operate differentially, buses are grouped by pairs as X and Y. If a given client uses a single ABUS (e.g. single-ended ADC), X and Y are just labels to differentiate the two buses.

When operating differentially, most APORT clients require that one input be chosen from an X bus and the other from a Y bus. For example, the ACMP block will not allow both positive and negative inputs to be chosen from X buses.

35.3.1 I/O Pin Considerations

For external analog signals routed through the APORT, the maximum supported analog I/O voltage will typically be limited to the MIN($V_{ANALOGSUPPLY}$, IOVDD) (where $V_{ANALOGSUPPLY}$ is the supply pin powering the analog module). Practically, this means that if IOVDD=1.8 V, the maximum supported analog IO voltage on APORT-routed signals will be limited to 1.8 V, regardless of the analog module supply voltage.

35.3.2 APORT ABUS Naming

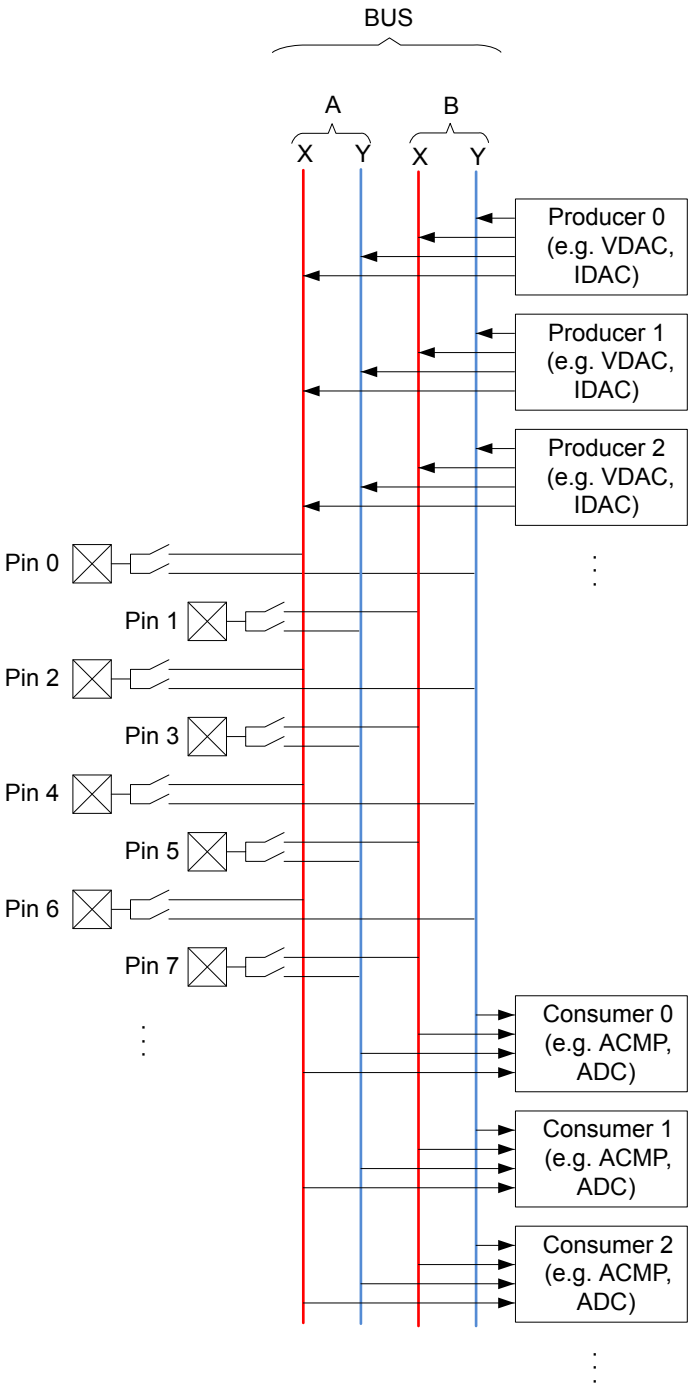


Figure 35.2. Conceptual APORT Structure

APORT ABUSes are prefixed with "BUS" and are grouped in pairs. Each pair is uniquely identified using a letter prefix ("A", "B", "C", etc.) followed by either a "X" or "Y" to identify the ABUS in the pair. For example, "BUSDX" decodes as: "BUS"=ABUS, "D"=pair, "X"=bus. Figure 35.2 Conceptual APORT Structure on page 1294 illustrates this organization.

APORT clients are generally described once in this reference manual regardless of its number of instances. For example, the ACMP client is described once, but the device could contain multiple instances of the ACMP. Because of this, for APORT client descriptions in this reference manual, the ABUS connections are generalized with the prefix "APORT" followed by a number (instead of the "BUS"

followed by a letter). It is possible that different instances of an APORT client connect to different ABUSes. For example, ACMP0 APORT1X might connect to the ABUS BUSAX while ACMP1 APORT1X might connect to ABUS BUSCX. Refer to the APORT Client Map in the device data sheet to map the generalized APORT client bus name to an actual device ABUS. A given ABUS has multiple switches which need to be identified. The switches on a bus are specified with the ABUS connection ID followed by a channel ID. For example, channel switch 7 on a given APORT client might be given as APORT1XCH7. Not all APORT channels map to actual GPIO. Refer to the APORT Client Maps in the device data sheet for APORT to GPIO mapping.

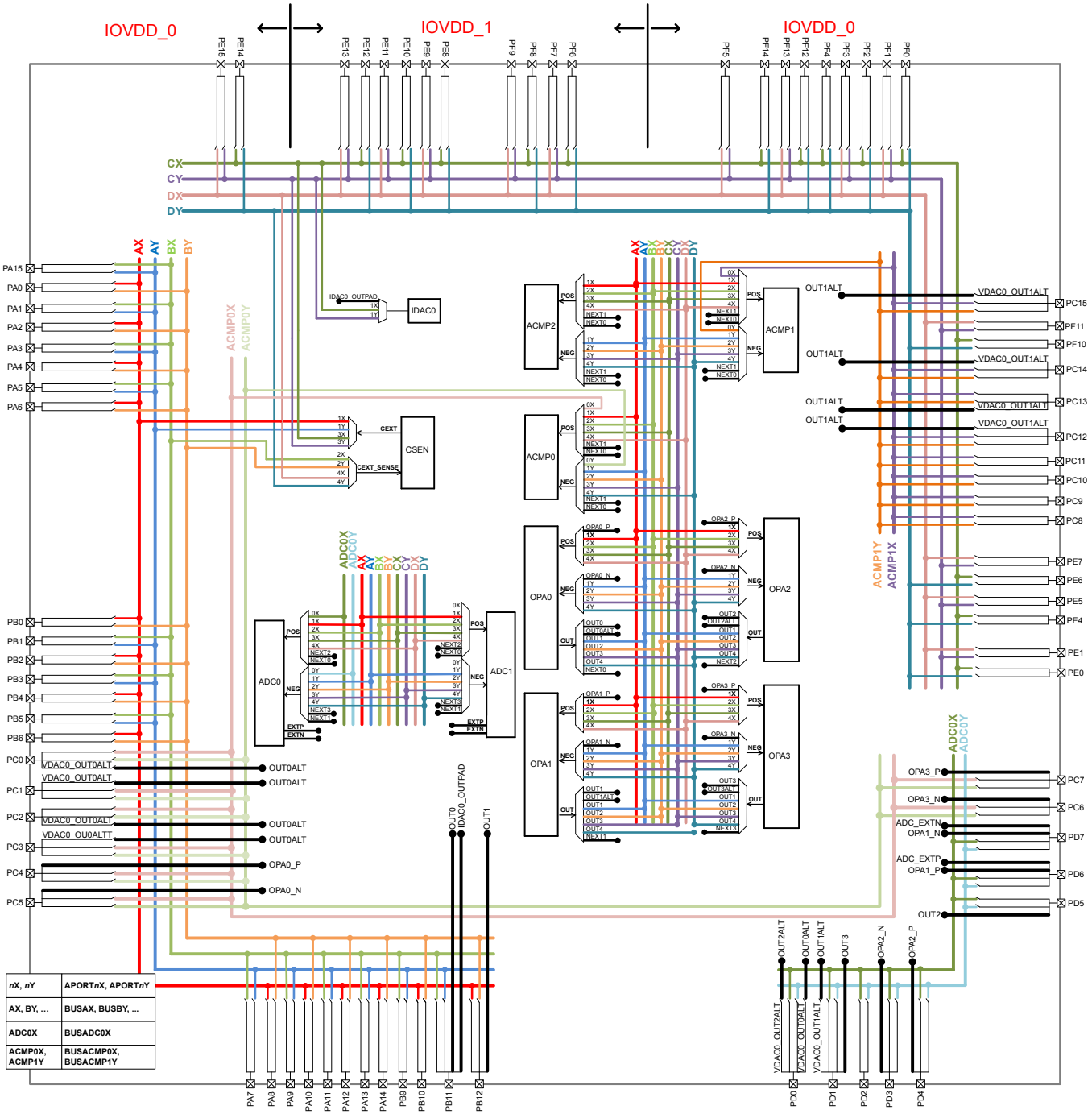


Figure 35.3. Detailed APORT Structure

Figure 35.3 Detailed APORT Structure on page 1295 shows all the possible routes between different peripherals and different pins via APORT BUS for the largest package of the EFM32GG12 device family. Note that, in the figure, the BUSxX and BUSxY are annotated as xX and xY, where x=A,B,C,D and the APORTnX and APORTnY are annotated as nX and nY, where n=1,2,3,4.

For example, the IDAC APORT output 1X can be routed to pin PF2 through BUSCX. The configuration required for this routing is as follows:

- Set IDAC_CTRL_APORTOUTSEL = APORT1XCH18. This selects the IDAC APORT output 1X and pin PF2.
- Set IDAC_CTRL_APORTOUTEN = 1 and IDAC_CTRL_APORTOUTENPRS = 0. This enables the IDAC to ungate it's output to BUSCX.

Another example, when ADC is configured to operate in single channel mode for differential inputs (see 28.3.3.1 Single Channel Mode for how to configure ADC in single channel mode), the positive ADC APORT input 2X and the negative ADC APORT input 2Y can be routed to pin PB5 and PB6 via BUSBX and BUSBY respectively with the following configuration:

- Set ADCn_SINGLECTRL_POSSEL = APORT2XCH21. This selects the pin PB5 for the positive input to the ADC.
- Set ADCn_SINGLECTRL_NEGSEL = APORT2YCH22. This selects the pin PB6 for the negative input to the ADC.

For smaller packages, not all GPIO pins are available. See the pinout sections of the device data sheet for pin availability on a specific device.

35.3.3 Managing ABUSes

The ABUSes of an APORT are shared resources. The user needs to be mindful of this in assigning I/O for different clients throughout the chip, as it is possible to have conflicts for a given ABUS. Each ABUS has an arbiter responsible for limiting the control over the ABUS to one and only one client. If multiple clients attempt to control an ABUS, the arbiter allows no client control over the ABUS and asserts a conflict signal to the clients. The user has the ability to check for such a conflict in each client's status, as well as generate an interrupt.

Having only one client control an ABUS is not the same as having only one user of an ABUS. It is possible for multiple clients to access a single ABUS, but requires all but one client to relinquish control of the ABUS. To do this, some clients have bits to disable bus mastership which are 0 by default. One example is the APORTXMASTERDIS bit in the ACMPn_CTRL. When set to 1, the client will not assert control of the APORT X BUS switches, but may still connect to an APORT X BUS that is controlled by another client.

For example, the ADC and ACMP both want to use the same pin on a particular ABUS the user might set the bus master disable bit to 1 for the ACMP. The ADC is the sole master of the switch configuration on that ABUS, so switches are configured using the configuration set in the ADC. When the ACMP channel is chosen on that same bus, the actual pin connection is dictated by the ADC settings for that bus.

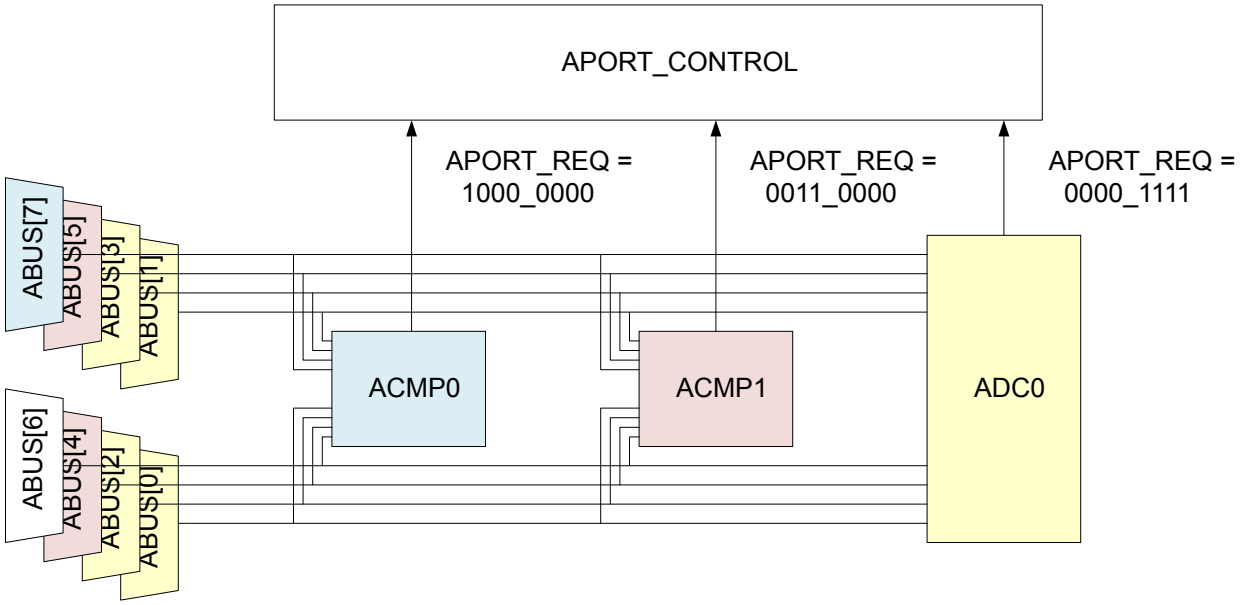


Figure 35.4. APORT Example 1

Figure 35.4 APORT Example 1 on page 1297 illustrates the sharing of APORT. For illustration purposes, each ABUS is identified by a numeric index (instead of BUSAX, BUSAY, BUSBX, etc.). Also, the requests from all the APORT clients are packed into a bit-vector named APORT_REQ to illustrate the request from the APORT Clients (instead of by name such as APORT1XREQ, APORT1YREQ, APORT2XREQ, etc.). In Figure 35.4 APORT Example 1 on page 1297, ABUS and client are the same color if the client has been granted the ABUS.

In Figure 35.4 APORT Example 1 on page 1297 ADC0 has requested ABUS[3:0], ACMP1 has requested ABUS[5:4], ACMP0 has requested ABUS[7], and ABUS[6] is unused. No APORT Client has requested the same ABUS as another, so there is no conflict.

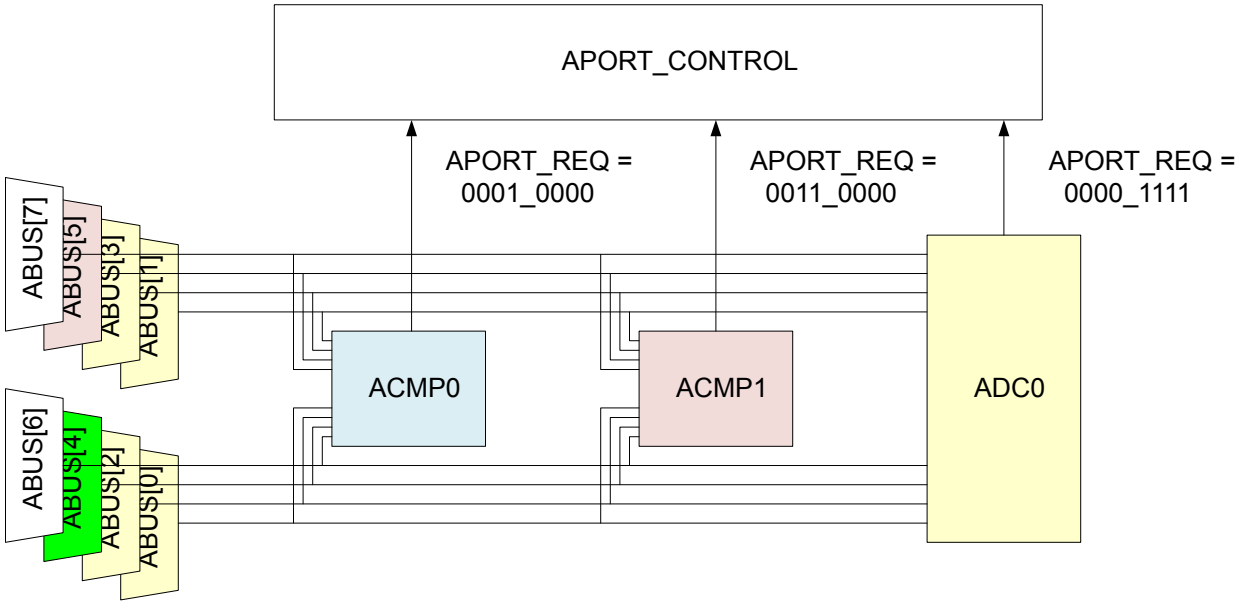


Figure 35.5. APORT Example 2: Bus Conflict

In Figure 35.5 APORT Example 2: Bus Conflict on page 1298 is a similar example to Figure 35.4 APORT Example 1 on page 1297, but now both ACMP0 and ACMP1 are requesting ABUS[4]. This is a configuration error, so APORT grants neither client ABUS[4]. The user must resolve the conflict before ABUS[4] is useable.

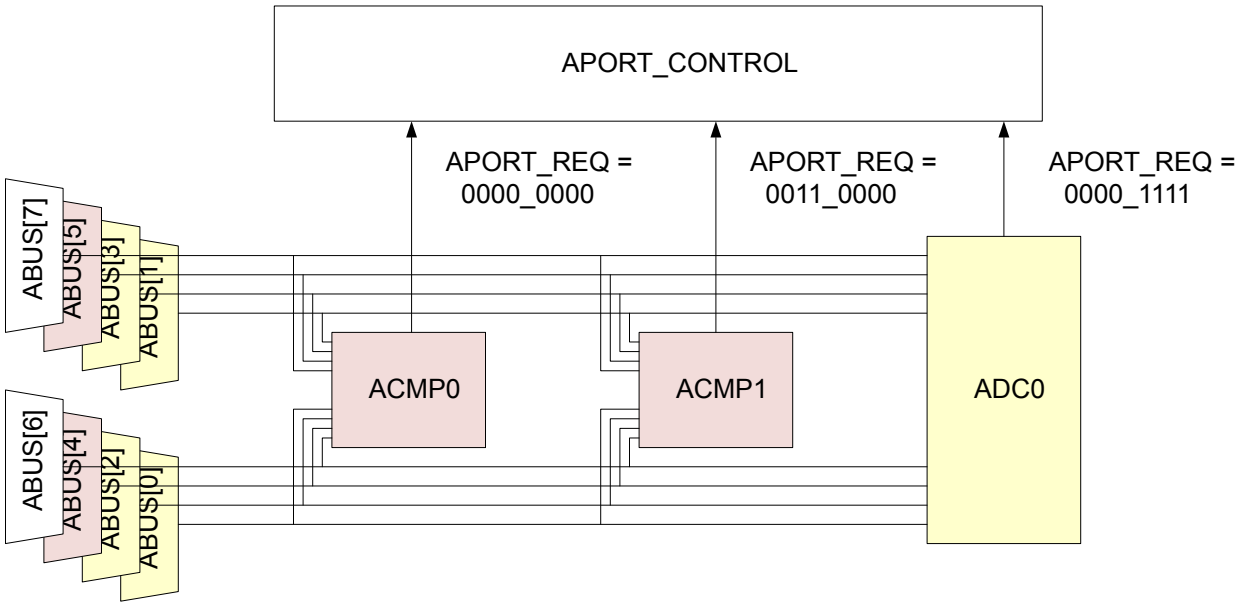
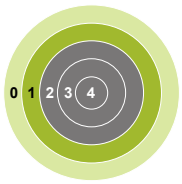
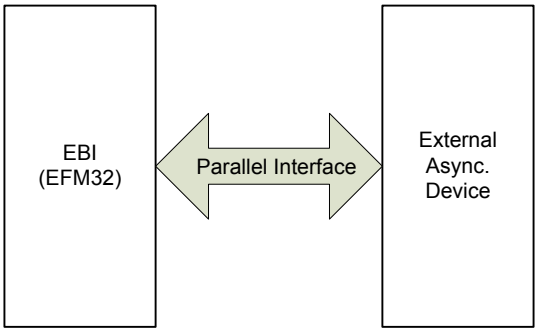


Figure 35.6. APORT Example 3: Sharing an ABUS

Figure 35.6 APORT Example 3: Sharing an ABUS on page 1298 illustrates ABUS sharing. Both ACMPs are configured identically, except ACMP0 has its APORTXMASTERDIS bit-field set to 1. There is only one APORT master for ABUS[5:4] in this case, so there is no conflict.

36. EBI - External Bus Interface





Quick Facts

What?

The EBI is used for accessing external parallel devices. The devices appear as a part of the EFM32 Giant Gecko 12's internal memory map and are therefore extremely simple to use.

Why?

Even though the EFM32 Giant Gecko 12 is versatile, there might be a need for specific external devices such as extra RAM, FLASH, LCD, TFT. The EBI simplifies the access to such devices.

How?

Through memory mapping the devices appear as a part of the internal memory map. When the processor performs read or writes to the address range of the EBI, the EBI handles the data transfers to and from the external devices. The EBI may be interfaced by the DMA, thus enabling operation in EM1 Sleep.

36.1 Introduction

The External Bus Interface provides access to external parallel interface devices such as SRAM, FLASH, ADCs and LCDs. The interface is memory mapped into the address bus of the Cortex-M4. This enables seamless access from software without manually manipulating the IO settings each time a read or write is performed. The data and address lines can be multiplexed in order to reduce the number of pins required to interface the external devices. The bus timing is adjustable to meet specifications of the external devices. The interface is limited to asynchronous devices and TFT.

36.2 Features

- Programmable interface for various memory types
  - 4 memory bank regions
  - Individual chip select line (EBI_CSn) per memory bank
  - Accurate control of setup, strobe, hold and turn-around timing per memory bank
  - Individual active high / active low setting of interface control signals per memory bank
  - Slave read/write cycle extension per memory bank
  - Page mode read
  - NAND Flash support
- Both multiplexed and non-multiplexed address and data line configurations
  - Up to 28 address lines
  - Up to 16-bit data bus width
- Automatic translation when AHB transaction width and memory width differ
- Configurable prefetch from external device
- Write buffer to limit stalling of the Cortex-M4 or DMA
- TFT Direct Drive
  - Programmable display and porch sizes
  - Programmable bus timing (frequency, setup and hold timing)
  - Individual active high / active low setting of interface control signals
  - Frame buffer can be either on-chip or off-chip
  - Alpha-blending and masking

36.3 Functional Description

An overview of the EBI module is shown in Figure 36.1 EBI Overview on page 1301. The EBI module consists of two submodules. The first submodule implements a generic external device interface to for example SRAM or Flash devices. The second submodule implements a TFT RGB interface which can be used together with the generic external device interface to perform TFT Direct Drive from an external framebuffer to a TFT display.

The EBI has multiplexed and non-multiplexed addressing modes. Fastest operation is achieved when using a non-multiplexed addressing mode. The multiplexed addressing modes are somewhat slower and require an external latch, but they use a significantly lower number of pins. The use of the 16 EBI_AD pin connections depends on the addressing mode. They are used for both address and data in the multiplexed modes. Also for the non-multiplexed 8-bit address mode both the address and data fit into these 16 EBI_AD pins. If more address bits or data bits are needed, external latches can be used to support up to 24-bit addresses or 16-bit data in the multiplexed addressing modes using only the 16 EBI_AD pins. Furthermore, independent of the addressing mode, up to 28 non-multiplexed address lines can be enabled on the EBI_A pin connections.

When a read operation is requested by the Cortex-M4 or DMA via the EBI's AHB interface, the address is transferred onto the EBI_AD and/or EBI_A bus. After a specific number of cycles, the EBI_REn pin is activated and data is read from the EBI_AD bus. When a write operation is requested, the address is transferred onto the EBI_AD and/or EBI_A bus and subsequently the write data is transferred onto the EBI_AD bus as the EBI_WEn pin is activated. The detailed operation in the supported modes is presented in the following sections.

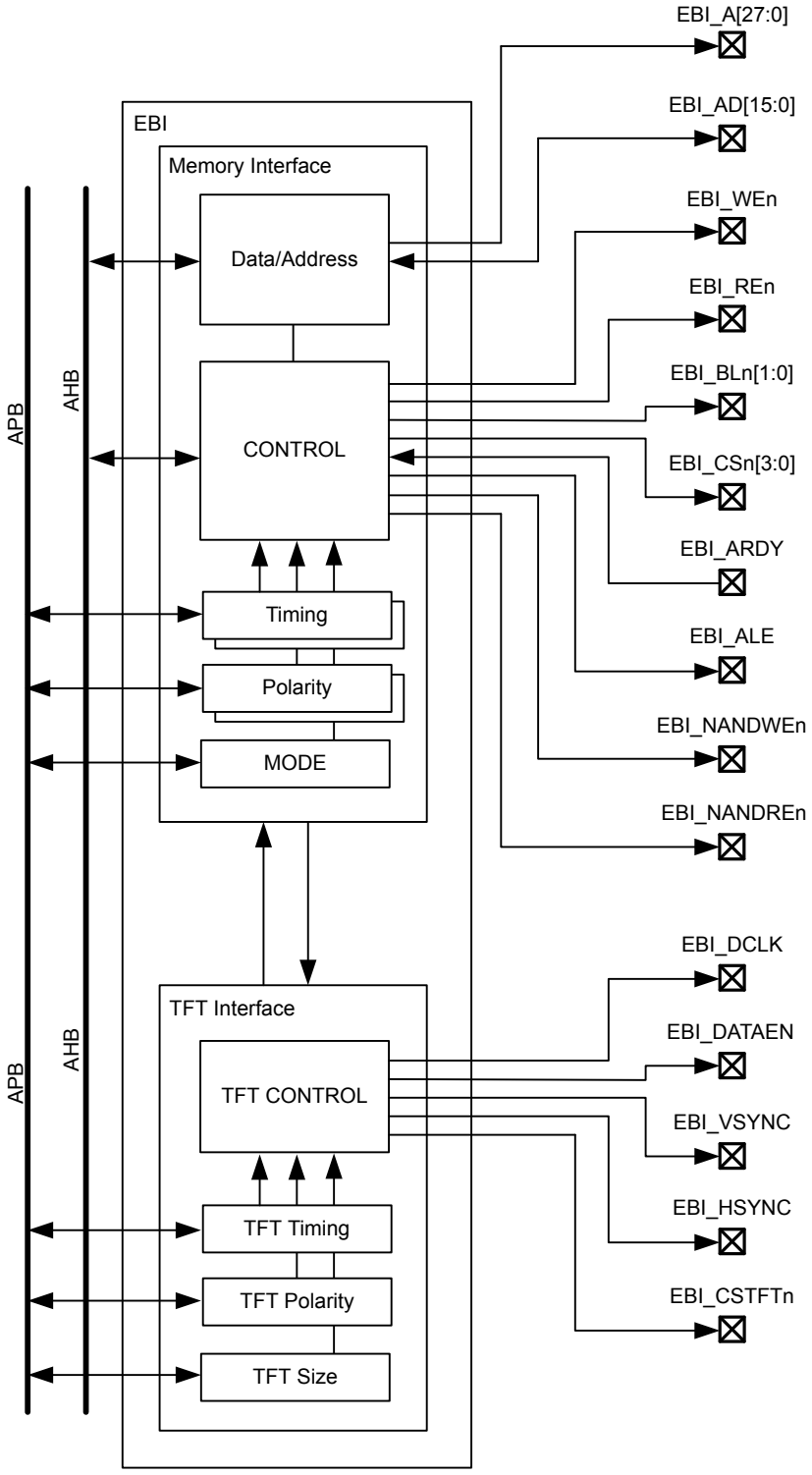


Figure 36.1. EBI Overview

36.3.1 Non-multiplexed 8-bit Data, 8-bit Address Mode

In this mode, 8-bit address and 8-bit data is supported. The address is put on the higher 8 bits of the EBI_AD lines while the data uses the lower 8 bits. This mode is set by programming the MODE field in the EBI_CTRL register to D8A8. The address space can be extended to 256 MB by using the EBI_A lines as described in 36.3.6 Extended Addressing. Read and write signals in 8-bit mode are shown in Figure 36.2 EBI Non-multiplexed 8-bit Data, 8-bit Address Read Operation on page 1302 and Figure 36.3 EBI Non-multiplexed 8-bit Data, 8-bit Address Write Operation on page 1302 respectively.

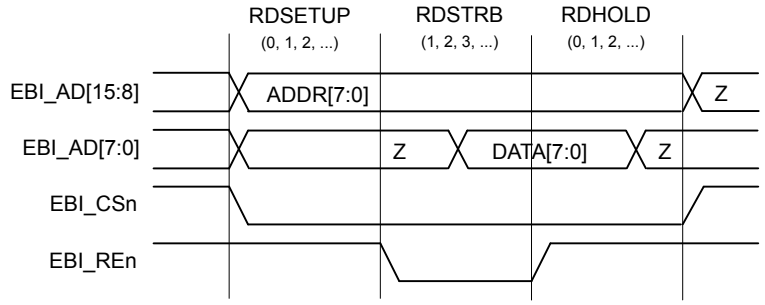


Figure 36.2. EBI Non-multiplexed 8-bit Data, 8-bit Address Read Operation

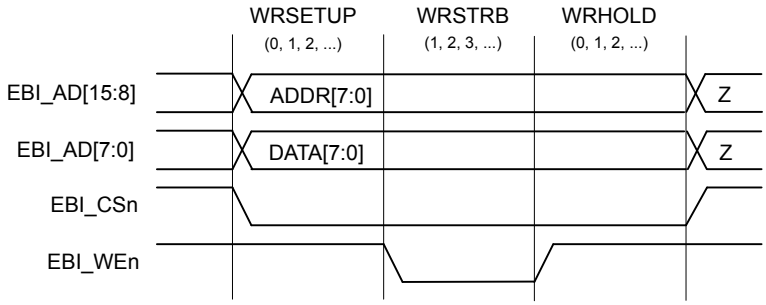


Figure 36.3. EBI Non-multiplexed 8-bit Data, 8-bit Address Write Operation

36.3.2 Multiplexed 16-bit Data, 16-bit Address Mode

In this mode, 16-bit address and 16-bit data is supported, but the utilization of an external latch is required. The 16-bit address and 16-bit data bits are multiplexed on the EBI_AD lines. An illustration of such a setup is shown in Figure 36.4 EBI Address Latch Setup on page 1303. This mode is set by programming the MODE field in the EBI_CTRL register to D16A16ALE.

Note: In this mode the 16-bit address is organized in 2-byte chunks at memory addresses aligned to 2-byte offsets. Consequently, the LSB of the 16-bit address will always be 0. In order to double the address space, the 16-bit address is internally shifted one bit to the right so that the LSB of the address driven into the EBI_AD bus, i.e. the EBI_AD[0]-bit, corresponds to the second least significant bit of the address, i.e. ADDR[1]. At the external device, the LSB of the address must be tied either low or high in order to create a full address.

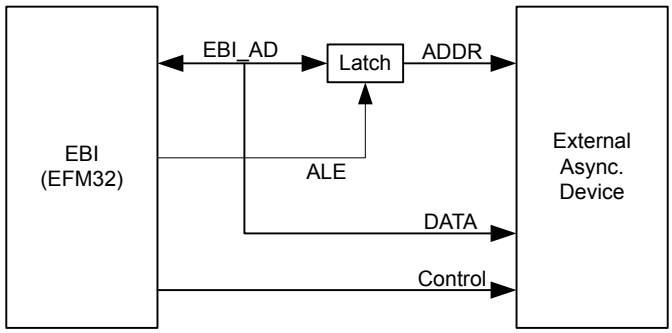


Figure 36.4. EBI Address Latch Setup

At the start of the transaction the address is output on the EBI_AD lines. The Latch is controlled by the ALE (Address Latch Enable) signal and stores the address. Then the data is read or written according to operation. Read and write signals are shown in Figure 36.5 EBI Multiplexed 16-bit Data, 16-bit Address Read Operation on page 1303 and Figure 36.6 EBI Multiplexed 16-bit Data, 16-bit Address Write Operation on page 1304 respectively.

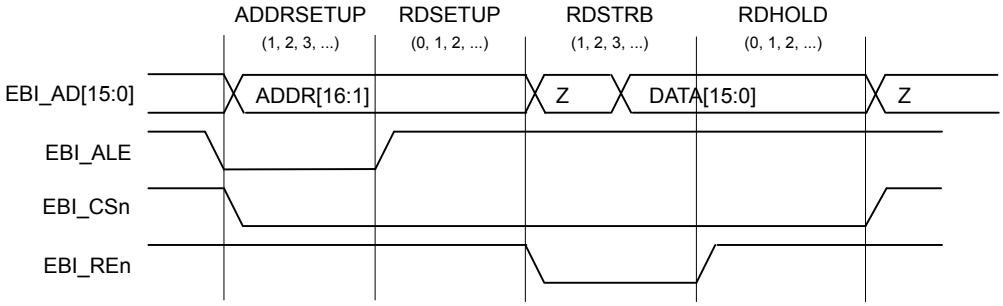


Figure 36.5. EBI Multiplexed 16-bit Data, 16-bit Address Read Operation

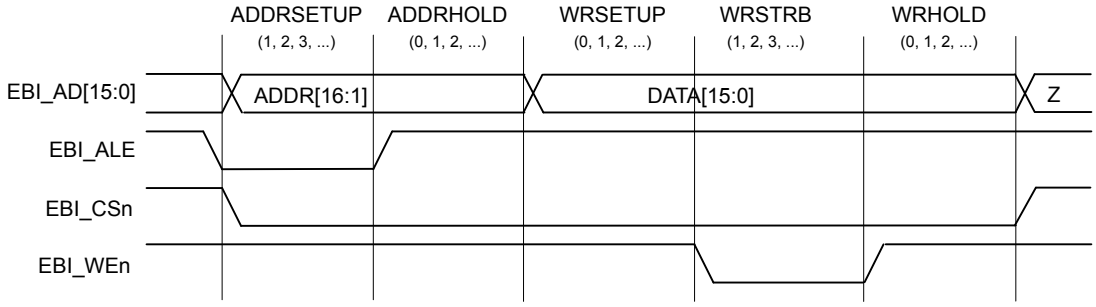


Figure 36.6. EBI Multiplexed 16-bit Data, 16-bit Address Write Operation

36.3.3 Multiplexed 8-bit Data, 24-bit Address Mode

This mode allows 24-bit address with 8-bit data multiplexed on the EBI_AD lines. The upper 8 bits of the EBI_AD lines are consecutively used for the highest 8 bits and the lowest 8 bits of the address. The lower 8 bits of the EBI_AD lines are used for the middle 8 address bits and for data. This mode is set by programming the MODE field in the EBI_CTRL register to D8A24ALE. Read and write signals are shown in Figure 36.7 EBI Multiplexed 8-bit Data, 24-bit Address Read Operation on page 1304 and Figure 36.8 EBI Multiplexed 8-bit Data, 24-bit Address Write Operation on page 1304 respectively.

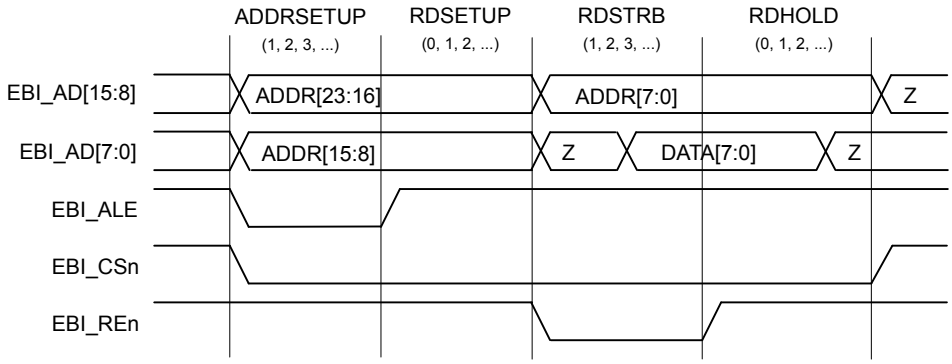


Figure 36.7. EBI Multiplexed 8-bit Data, 24-bit Address Read Operation

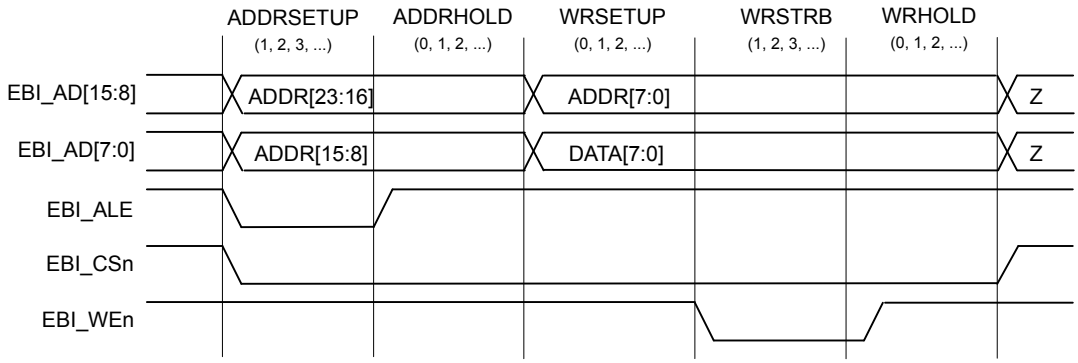


Figure 36.8. EBI Multiplexed 8-bit Data, 24-bit Address Write Operation

36.3.4 Non-multiplexed 16-bit Data, N-bit Address Mode

In this non-multiplexed mode 16-bit data is driven on the 16 EBI_AD lines. The addresses are driven on the EBI_A lines. The address space can be up to 256 MB as described in 36.3.6 Extended Addressing. This mode is set by programming the MODE field in the EBI_CTRL register to D16. Read and write signals are shown in Figure 36.9 EBI Non-multiplexed 16-bit Data Read Operation with Extended Address on page 1305 and Figure 36.10 EBI Non-multiplexed 16-bit Data Write Operation with Extended Address on page 1305 respectively for the case in which N address lines on EBI_A have been enabled.

Note: In this mode the 16-bit address is organized in 2-byte chunks at memory addresses aligned to 2-byte offsets. Consequently, the LSB of the 16-bit address will always be 0. In order to double the address space, the 16-bit address is internally shifted one bit to the right so that the LSB of the address driven into the EBI_A bus, i.e. the EBI_A[0]-bit, corresponds to the second least significant bit of the address, i.e. ADDR[1]. At the external device, the LSB of the address must be tied either low or high in order to create a full address.

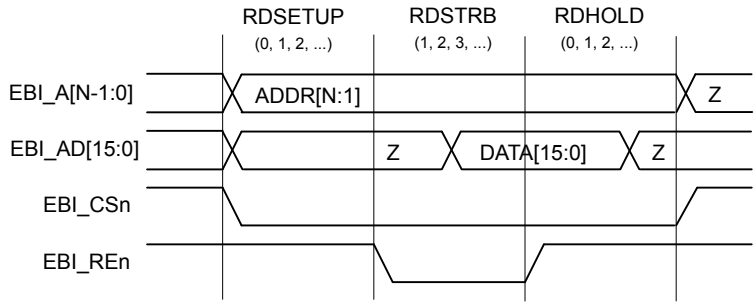


Figure 36.9. EBI Non-multiplexed 16-bit Data Read Operation with Extended Address

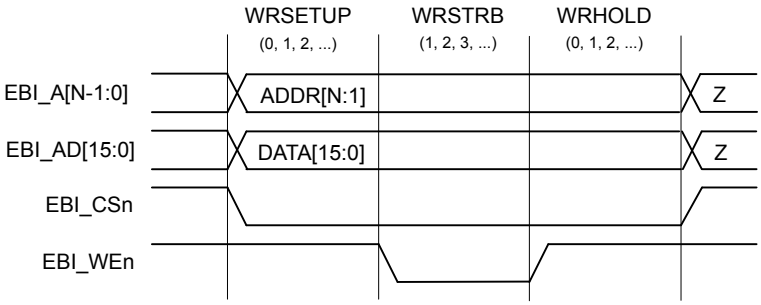


Figure 36.10. EBI Non-multiplexed 16-bit Data Write Operation with Extended Address

36.3.5 Page Mode Read Operation

Page mode read operation can enhance the performance of a sequence of consecutive asynchronous read transactions by allowing data at subsequent intrapage addresses to be read faster. Page mode operation is enabled by setting the PAGEMODE bitfield in the EBI_RDTIMING (or EBI_RDTIMINGn) register to 1. If enabled, the RDPA bitfield in the EBI_PAGECTRL register defines the duration of an intrapage access and the PAGELEN bitfield in the EBI_PAGECTRL register defines the number of members in a page. Page mode reads can for example be triggered by consecutive reads resulting from wide AHB reads which are automatically translated into multiple narrow external device reads. Page mode reads can also be triggered by sequential reads resulting from the EBI prefetch unit.

The number of members in a page together with the width of the external device and the INCHIT bit of the EBI_PAGECTRL register define whether an address change results in an interpage access or in an intrapage access as shown in Table 36.1 EBI Intrapage hit condition for read on address Addr (non-mentioned Addr bits are unchanged) on page 1306.

Table 36.1. EBI Intrapage hit condition for read on address Addr (non-mentioned Addr bits are unchanged)

PAGELEN, INCHIT	8-bit External Device	16-bit External Device
PAGELEN=MEMBER4, INCHIT=0	Addr[1:0] changed	Addr[2:0] changed
PAGELEN=MEMBER8, INCHIT=0	Addr[2:0] changed	Addr[3:0] changed
PAGELEN=MEMBER16, INCHIT=0	Addr[3:0] changed	Addr[4:0] changed
PAGELEN=MEMBER32, INCHIT=0	Addr[4:0] changed	Addr[5:0] changed
PAGELEN=MEMBER4, INCHIT=1	Addr[1:0] incremented by 1	Addr[2:0] incremented by 2
PAGELEN=MEMBER8, INCHIT=1	Addr[2:0] incremented by 1	Addr[3:0] incremented by 2
PAGELEN=MEMBER16, INCHIT=1	Addr[3:0] incremented by 1	Addr[4:0] incremented by 2
PAGELEN=MEMBER32, INCHIT=1	Addr[4:0] incremented by 1	Addr[5:0] incremented by 2

The initial page mode transaction uses the read setup and read strobe timing as shown in Figure 36.2 EBI Non-multiplexed 8-bit Data, 8-bit Address Read Operation on page 1302, Figure 36.5 EBI Multiplexed 16-bit Data, 16-bit Address Read Operation on page 1303, Figure 36.7 EBI Multiplexed 8-bit Data, 24-bit Address Read Operation on page 1304 or Figure 36.9 EBI Non-multiplexed 16-bit Data Read Operation with Extended Address on page 1305 depending on the used addressing mode. Subsequent transactions are started by changing the low-order address bits and use the page access time defined in the RDPA bitfield of the EBI_PAGECTRL register. The read hold state RDHOLD is only performed at the end of a page mode read sequence or when bus turn-around occurs. Note that bus turn-around can occur even if only read transactions are performed as the D16A16ALE addressing mode will drive the EBI_AD lines when programming the external address latch. In this case one bus turn-around RDHOLDX cycle is automatically inserted in between the read and the write action on the EBI_AD lines. Note that for the D16A16ALE addressing mode the RDPA state immediately follows the ADDRSETUP state, so the HALFALE feature will typically be required to satisfy the external address latch hold requirement. In the D8A24ALE addressing mode there is no need to reprogram the external address latch for intrapage addresses as the external latch then only latches the most significant, non-changed address lines. The following figures show typical page mode read sequences for all addressing modes.

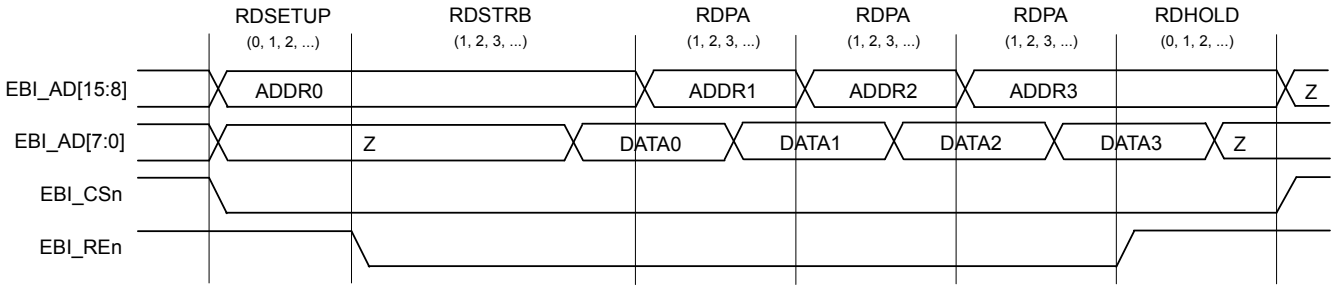


Figure 36.11. EBI Page Mode Read Operation for D8A8 addressing mode

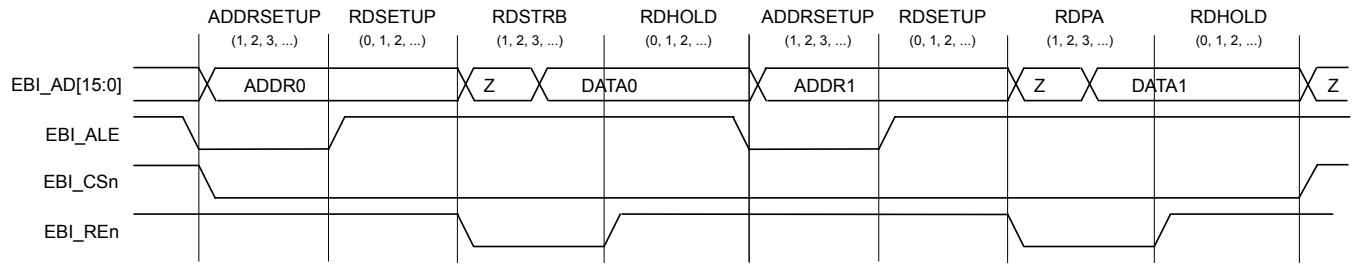


Figure 36.12. EBI Page Mode Read Operation for D16A16ALE addressing mode

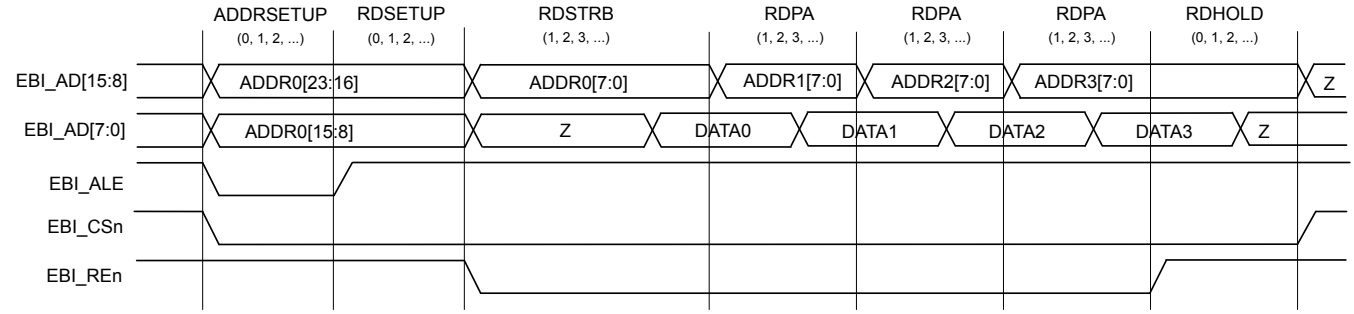


Figure 36.13. EBI Page Mode Read Operation for D8A24ALE addressing mode

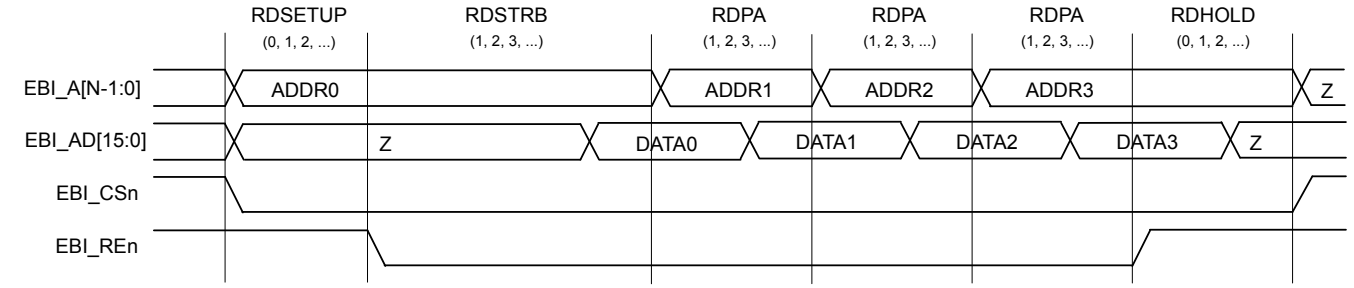


Figure 36.14. EBI Page Mode Read Operation for D16 addressing mode

The maximum duration that a page is kept open is defined in the KEEPOPEN bitfield of the EBI_PAGECTRL register. New read transactions which hit in an open page are started with RDPA intrapage timing if the KEEPOPEN time has not been exceeded at the start of such a transaction. The default setting of KEEPOPEN, which is equal to 0, will therefore never allow for intrapage timing to occur. Transactions are allowed to finish if the KEEPOPEN time is exceeded during the transaction. Otherwise the RDSTRB interpage timing is used for the read transaction. Next to exceeding the KEEPOPEN time there are other reasons for closing an open page. In particular EBI transactions which result in a write or a non-intrapage read always cause the page to be closed. Also the lack of a new EBI transaction will cause an open page to be closed. In order to prevent this last scenario as much as possible read transactions can often be made back to back. This is achieved by enabling prefetching by setting PREFETCH to 1 in the EBI_RDTIMING (or EBI_RDTIMINGn) register and by disallowing idle state insertion in between transfers by setting the NOIDLE (or NOIDLEn) bit to 1 in EBI_CTRL register. Figure 36.15 EBI Page Closing on page 1308 shows an example in which only ADDR1 benefits from intrapage timing because an unrelated AHB transfer not directed at the EBI causes late arrival of ADDR2. ADDR2 arrives too late to be inserted as a back to back read transfer. The page is considered closed and ADDR2 can therefore not benefit from intrapage timing and it results in an interpage access instead.

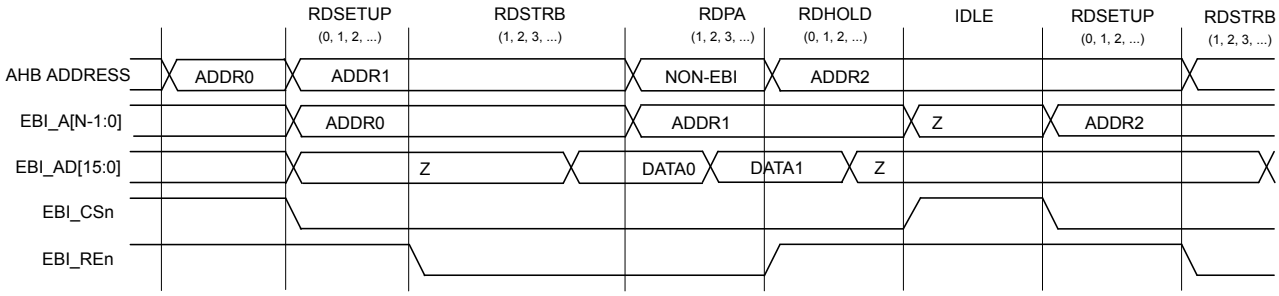


Figure 36.15. EBI Page Closing

36.3.6 Extended Addressing

Extended addressing is used to extend the address range for any of the addressing modes described in 36.3.4 Non-multiplexed 16-bit Data, N-bit Address Mode, 36.3.1 Non-multiplexed 8-bit Data, 8-bit Address Mode , 36.3.2 Multiplexed 16-bit Data, 16-bit Address Mode and 36.3.3 Multiplexed 8-bit Data, 24-bit Address Mode . Up to 28 address bits can be individually enabled on the EBI_A address lines providing up to 256 MB of address space per memory bank. The operation on the EBI_AD lines is not affected by this. See 36.3.12 Bank Access for the memory map definitions related to the EBI. An example of address extension for the D16 mode is shown in Figure 36.9 EBI Non-multiplexed 16-bit Data Read Operation with Extended Address on page 1305 and Figure 36.10 EBI Non-multiplexed 16-bit Data Write Operation with Extended Address on page 1305. A further example for address extension in the multiplexed 16-bit data, 16-bit address mode of 36.3.2 Multiplexed 16-bit Data, 16-bit Address Mode is shown in Figure 36.16 EBI Extended Address Latch Setup on page 1309. This is achieved by programming the MODE field in the EBI_CTRL register to D16A16ALE and by enabling the required address lines via the ALB and APEN bitfields of the EBI_ROUTEPEN register.

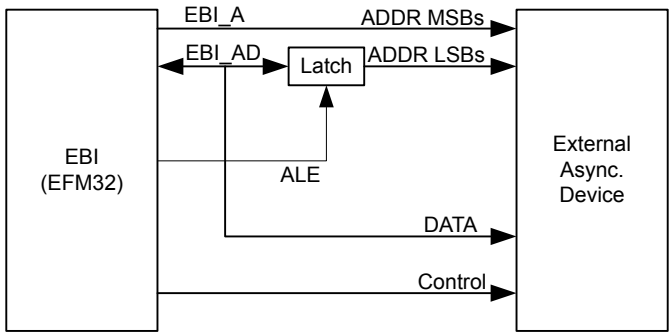


Figure 36.16. EBI Extended Address Latch Setup

Read and write signals for using extended addressing in the D16A16ALE mode are shown in Figure 36.17 EBI 16-bit Data Multiplexed Read Operation using Extended Addressing on page 1309 and Figure 36.18 EBI 16-bit Data Multiplexed Write Operation using Extended Addressing on page 1310 respectively for the case in which N extra address lines have been enabled. At the start of the transaction the lower address bits are output on the EBI_AD lines. The Latch is controlled by the ALE (Address Latch Enable) signal and stores the address. Then the data is read or written according to operation. The higher address bits are output on the EBI_A lines throughout the transfer.

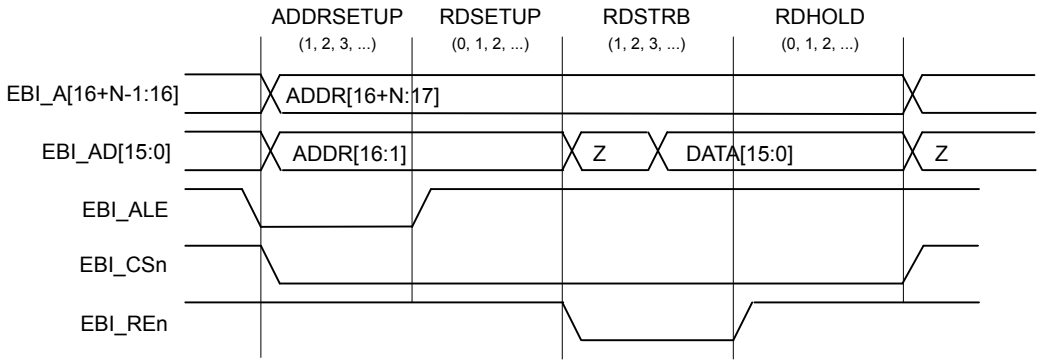


Figure 36.17. EBI 16-bit Data Multiplexed Read Operation using Extended Addressing

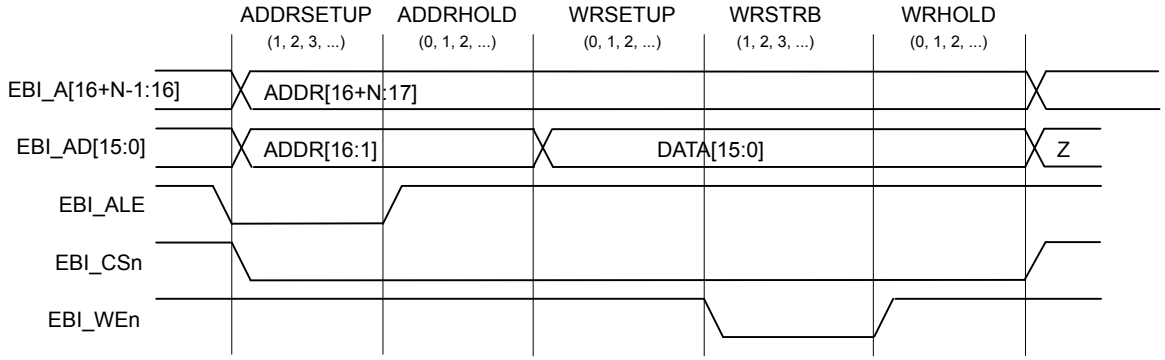


Figure 36.18. EBI 16-bit Data Multiplexed Write Operation using Extended Addressing

In order to minimize the pin requirements both the lower bound and the upper bound of the enabled EBI_A lines can be set. This is done in the ALB and APEN bitfields of the EBI_ROUTEPEN register respectively. For example, in case all memory banks use the 8-bit addressing mode D8A8, then the lower 8 address bits are always output on EBI_AD. Therefore, if address extension is required, only address bits 8 and upwards need to be enabled on EBI_A. This is done by setting the EBI_A lower bound to 8 by setting ALB to A8 in EBI_ROUTEPEN and by enabling the required higher address lines via the APEN bitfield in EBI_ROUTEPEN. The operation of the APEN and ALB bitfields is shown in Table 36.2 EBI Enabling EBI_ADDR lines for transaction with address Addr and data Data on page 1310 for some typical configurations.

Table 36.2. EBI Enabling EBI_ADDR lines for transaction with address Addr and data Data

Configuration	Addresses on EBI_A	Addresses/data on EBI_AD
MODE = D8A8, ALB = A8, APEN = A28	EBI_A[27:8] = Addr[27:8]	EBI_AD[15:0] = {Addr[7:0], Data[7:0]}
MODE = D16A16ALE, ALB = A16, APEN = A27	EBI_A[26:16] = Addr[27:17]	EBI_AD[15:0] = Addr[16:1]; Data[15:0]
MODE = D8A24ALE, ALB = A24, APEN = A28	EBI_A[27:24] = Addr[27:24]	EBI_AD[15:0] = Addr[23:8]; {Addr[7:0], Data[7:0]}
MODE = D16, ALB = A0, APEN = A27	EBI_A[26:0] = Addr[27:1]	EBI_AD[15:0] = Data[15:0]

36.3.7 Prefetch Unit and Write Buffer

Prefetching from external memory can enhance the performance of a sequence of consecutive transfers. In particular sequential code execution from external memory can benefit from prefetch. Also prefetch will typically lead to better utilization of intrapage accesses in case page mode is used. If prefetch is enabled, the prefetch unit will sequentially prefetch one data item of the same width as the last Cortex-M4 or DMA read transaction handled by the EBI. Note that one prefetch transaction might lead to multiple external device transactions as described in Table 36.3 EBI Mapping of AHB Transactions to External Device Transactions on page 1315. Prefetch is not performed in reaction to write transactions, nor will prefetch cross bank boundaries. The prefetch unit is enabled via the PREFETCH bitfield in the EBI_RDTIMING and EBI_RDTIMINGn registers. When the ITS bitfield in the EBI_CTRL register is set to 0, the PREFETCH bitfield from EBI_RDTIMING applies to all 4 memory banks. When ITS is set to 1 the prefetch unit can be individually enabled per bank. In this case register EBI_RDTIMING only applies to bank 0. Prefetch enabling for bank n is then defined in the EBI_RDTIMINGn register.

The EBI has a 1 entry 32-bit wide write buffer. The write buffer can be used to limit stalling by partially decoupling the Cortex-M4 or DMA from a potentially slow external device. Only writes which are guaranteed to not cause an error (e.g. timeout) in the EBI will be buffered when the write buffer is enabled, such that precise error generation is guaranteed. The write buffer is disabled via the WBUFDIS bitfield in the EBI_WRTIMING and EBI_WRTIMINGn registers. When the ITS bitfield in the EBI_CTRL register is set to 0, the WBUFDIS bitfield from EBI_WRTIMING applies to all 4 memory banks. When ITS is set to 1 the write buffer can be individually disabled per bank. In this case register EBI_WRTIMING only applies to bank 0. Write buffer disabling for bank n is then defined in the EBI_WRTIMINGn register.

The AHBACT status bit in the EBI_STATUS register indicates whether an AHB transaction is still active in the EBI or not. When performing an AHB write, the AHBACT bit stays 1 until the required transaction(s) with the external device have finished, independent of whether the AHB write gets buffered or not. On an AHB read with prefetching enabled, AHBACT stays high until the potential external device prefetch transaction(s) have finished.

36.3.8 Strobe Length

For external devices with low, but non-zero, setup requirements the performance overhead for EBI transactions can be relatively large if a full cycle setup time needs to be used. It is possible to borrow half of the cycle time from a neighboring strobe phase in order to define setup times with a granularity of half the internal clock period.

The durations of the EBI_ALE, EBI_REn, EBI_WEn, EBI_NANDREn and EBI_NANDWEn strobes can be individually decreased by half the internal clock period via the HALFALE, HALFRE and HALFWE bitfields in the address timing, read timing and write timing registers respectively. In case of EBI_ALE the trailing edge of the strobe can be moved half a clock period earlier. In case of EBI_REn, EBI_WEn, EBI_NANDREn and EBI_NANDWEn the leading edge of the strobe can be moved half a clock period later. Decreasing the length of the EBI_ALE strobe can be thought of as increasing the length of the RDSETUP phase by the same amount. Similarly, decreasing the length of the EBI_REn, EBI_WEn, EBI_NANDREn, EBI_NANDWEn strobes can be thought of as increasing the length of the RDSETUP and WRSETUP phases. Note that the length of the ADDRSETUP, RDSTRB, and WRSTRB phases is still 1 or more internal clock cycles. For example, when HALFRE is set to 1 and RDSTRB is programmed to 2, the length of the RDSTRB phase is 2 cycles. The duration of the EBI_REn pulse is however decreased by half a cycle to 1 1/2 cycles.

Figure 36.5 EBI Multiplexed 16-bit Data, 16-bit Address Read Operation on page 1303 and Figure 36.6 EBI Multiplexed 16-bit Data, 16-bit Address Write Operation on page 1304 respectively show read and write transactions in the multiplexed 16-bit address, 16-bit data mode in which half strobes are enabled for EBI_ALE, EBI_REn and EBI_WEn.

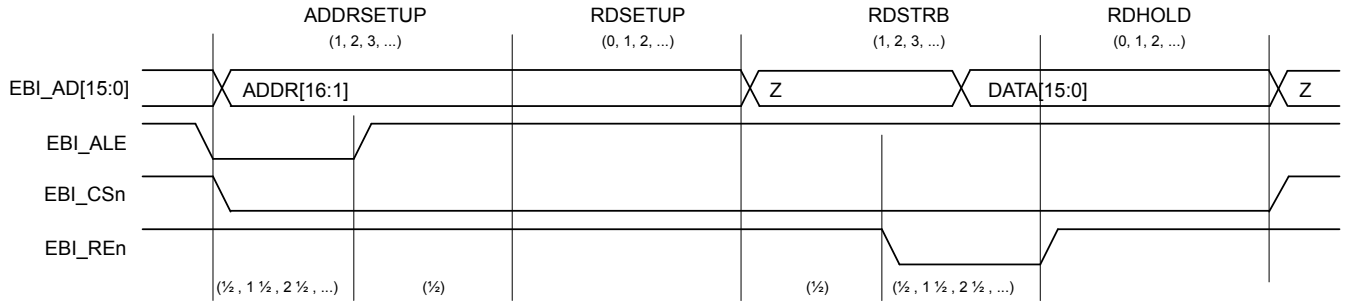


Figure 36.19. EBI Multiplexed Read Operation with Reduced Length Strobes

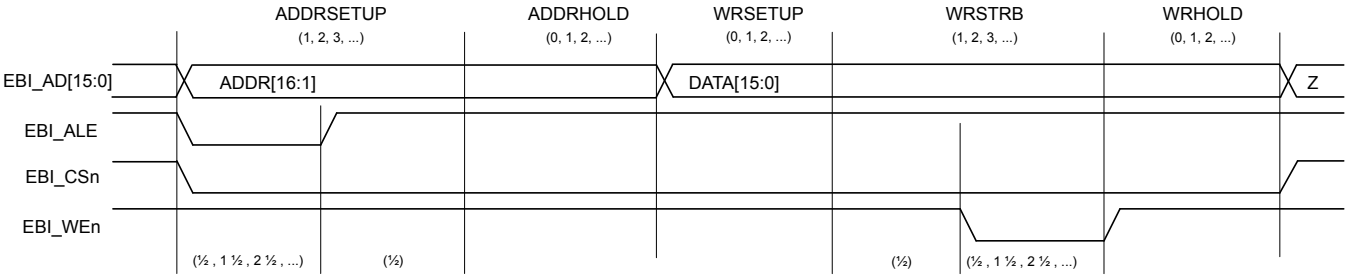


Figure 36.20. EBI Multiplexed Write Operation with Reduced Length Strobes

36.3.9 Bus Turn-around and Idle Cycles

The EBI_AD lines can be driven by either the EFM32 Giant Gecko 12 or by the external device. Depending on the characteristics of an external device, the RDHOLD should be programmed to ensure adequate bus turn-around time. Default the EBI inserts an initial IDLE cycle, during which the EBI does not drive the EBI_AD lines, after each external transaction. Furthermore, the EBI deasserts the EBI_CSn, EBI_REn, and EBI_WEn lines during IDLE cycles. In case of subsequent IDLE cycles, after the initial one, the EBI will drive the EBI_AD lines while keeping the EBI_CSn, EBI_REn, and EBI_WEn lines deasserted. The IDLE state insertion is shown for two back-to-back read transactions in Figure 36.21 EBI Enforced IDLE cycles between Transactions on page 1313. In case that the IDLE state provides the required bus turn-around time, the RDHOLD parameter can be programmed to 0. For increased performance, the automatic IDLE state insertion can be prevented by setting the NOIDLE/NOIDLEn bits in the EBI_CTRL register to 1. This scenario is shown in Figure 36.22 EBI No Enforced IDLE cycles between Transactions on page 1313 for two back-to-back reads in a non-multiplexed addressing mode. Note that in case RDSETUP and RDHOLD are both programmed to 0, then the EBI_REn line will not be deasserted between back-to-back read transfers. The same will happen for non-multiplexed back-to-back write transactions with WRSETUP and WRHOLD both programmed to 0. In case that NOIDLE/NOIDLEn is 1 and a read is immediately followed by a write on the EBI_AD lines, one bus turn-around cycle called RDHOLDX is automatically inserted in between the read and the write action. During a RDHOLDX cycle the external EBI signals are driven in the same way as during regular RDHOLD cycles, i.e. the EBI_REn line will get deasserted while the EBI_CSn line will stay asserted.

An IDLE cycle will automatically get inserted for the following cases:

- Between two external device transactions in case the NOIDLE/NOIDLEn bit is 0.
- Between two external device transactions to different banks.
- When no request for an external transaction is available in the EBI.

A RDHOLDX cycle will automatically get inserted for the following case:

- Between a read and a subsequent write on the EBI_AD lines. Note that this is only possible if NOIDLE/NOIDLEn is set to 1. Also note that a read in a multiplexed addressing mode (e.g. D16A16ALE) starts with a write on the EBI_AD lines when it is in the ADDRSETUP state.

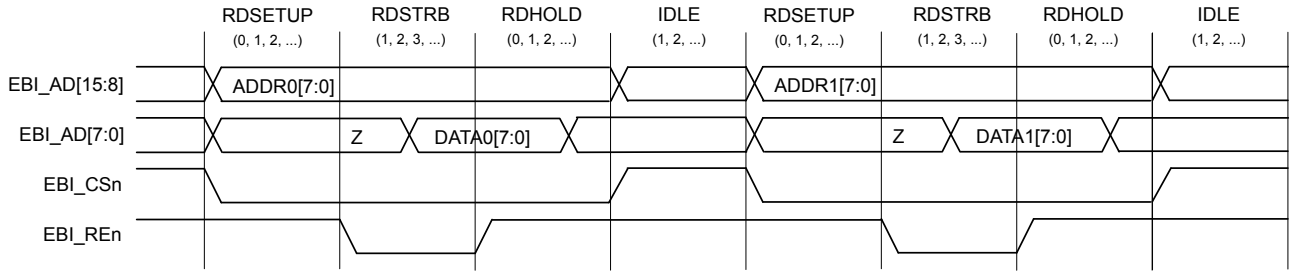


Figure 36.21. EBI Enforced IDLE cycles between Transactions

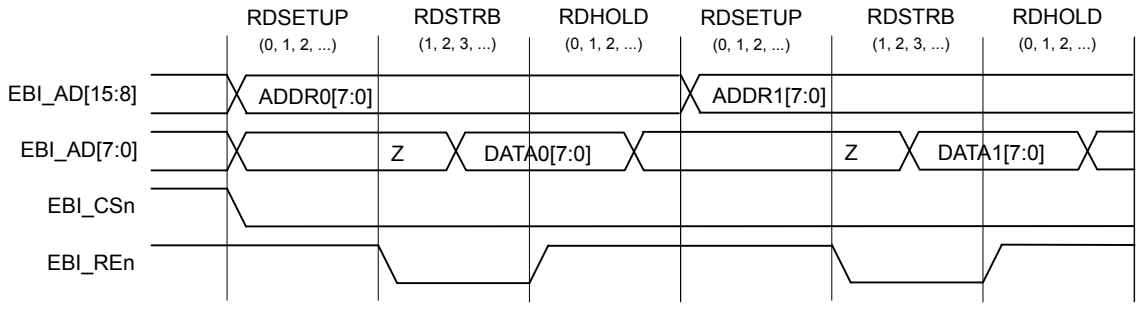


Figure 36.22. EBI No Enforced IDLE cycles between Transactions

Note: In case NOIDLE/NOIDLEn bits are set in EBI_CTRL the read or write strobes can remain asserted for back-to-back transfers if no further separation is guaranteed via for example RDSETUP, RDHOLD, WRSETUP, or WRHOLD bitfields.

36.3.10 Timing

The duration of the states in the transaction is defined by the corresponding uppercase name above the state, e.g. the address setup state in Figure 36.8 EBI Multiplexed 8-bit Data, 24-bit Address Write Operation on page 1304 is active for a number of internal clock cycles defined by ADDRSET bitfield in the EBI_ADDRTIMING register. Similar timing can be defined by the RDSTRB bitfield in the EBI_RDTIMING register and WRSTRB in the EBI_WRTIMING register. These parameters all have a minimum duration of 1 cycle, which is set by HW in case the bitfield is programmed to 0.

The setup and hold timing parameters are ADDRHOLD in the EBI_ADDRTIMING register, RDHOLD and RDSETUP in the EBI_RDTIMING register and WRHOLD and WR SETUP in the EBI_WRTIMING register. Writing a value m to one of these bitfields results in a duration of the corresponding state of m cycles. If these parameters are set to 0, it effectively means that the state is skipped.

Page mode access time is defined in the RDPA bitfield of the EBI_PAGECTRL register. This parameters has a minimum duration of 1 cycle, which is set by HW in case the bitfield is programmed to 0.

When the ITS bitfield in the EBI_CTRL register is set to 0, the timing set defined in the EBI_ADDRTIMING, EBI_RDTIMING and EBI_WRTIMING registers applies to all 4 memory banks. When ITS is set to 1 each memory bank uses an individual timing set. In this case registers EBI_ADDRTIMING, EBI_RDTIMING and EBI_WRTIMING only apply to bank 0. Timing for bank n is then defined in the EBI_ADDRTIMINGn, EBI_RDTIMINGn and EBI_WRTIMINGn registers.

Note: All timing related bitfields have a default value which is equal to the highest possible value for these bitfields, which makes the default values a better fit for slow memory devices. This differs from the EFM32G devices in which the default values correspond to the lowest possible values, which would only be appropriate for fast memory devices.

36.3.11 Data Access Width

The mapping of AHB transactions to external device accesses depends on the data width of the external device and on whether or not it supports byte lanes. The data width of external devices is specified in the MODE and MODEn bitfields of the EBI_CTRL register. An external device is specified to be either 8-bit or 16-bit wide. Availability of byte lane support by the external device is specified via the BL and BLn bitfields of the EBI_CTRL register. When the ITS bitfield in the EBI_CTRL register is set to 0, the MODE and BL bitfields apply to all 4 memory banks. When ITS is set to 1 each memory bank uses an individual mode and byte lane enable definition. In this case bitfields MODE and BL only apply to bank 0. The mode and byte lane availability for bank n is then defined in the MODEn and BLn bitfields.

In case the AHB transaction width does not match the width of the selected device, the EBI automatically translates the AHB transaction into 1 or more external device transactions matching the capabilities of that device. If one AHB transaction is translated into multiple external transactions, then the external transactions have incrementing addresses and start with the lowest data byte(s) from the AHB transaction. The translation, and possibly bus fault generation, is explained below and in Table 36.3 EBI Mapping of AHB Transactions to External Device Transactions on page 1315:

- If the AHB transaction width is larger than the external device width, then multiple consecutive external transactions are performed starting with the least significant data.
- If the AHB transaction width is smaller than the external device width, then EBI behavior depends on whether or not byte lanes are available for the selected device. Reads either use byte lane support when available, or read according to the full external device width and disregard the superfluous data. Writes normally either use byte lane support when available, or perform a read-modify-write sequence to only change the required data. However, NAND Flash does not support byte lanes or random access read-modify-write and therefore a hard fault is generated in case of an 8-bit write to a bank designated as 16-bit NAND bank.

Table 36.3. EBI Mapping of AHB Transactions to External Device Transactions

Data Access by Cortex-M4, DMA, or prefetch	8-bit External Device (non-NAND) transaction(s)	16-bit External Device (non-NAND) transaction(s)(with byte lanes)	16-bit External Device (non-NAND) transaction(s)(without byte lanes)	8-bit NAND Flash transaction(s)	16-bit NAND Flash transaction(s)
8-bit read	1 x 8-bit read	1 x 8-bit read (using byte lane)	1 x 16-bit read	1 x 8-bit read	1 x 16-bit read
16-bit read	2 x 8-bit read	1 x 16-bit read	1 x 16-bit read	2 x 8-bit read	1 x 16-bit read
32-bit read	4 x 8-bit read	2 x 16-bit read	2 x 16-bit read	4 x 8-bit read	2 x 16-bit read
8-bit write	1 x 8-bit write	1 x 8-bit write (using byte lane)	1 x 16-bit read; 1 x 16-bit write (read-modify-write)	1 x 8-bit write	- (Hard fault)
16-bit write	2 x 8-bit write	1 x 16-bit write	1 x 16-bit write	2 x 8-bit write	1 x 16-bit write
32-bit write	4 x 8-bit write	2 x 16-bit write	2 x 16-bit write	4 x 8-bit write	2 x 16-bit write

36.3.12 Bank Access

The EBI is split in 4 different address regions, each connected to an individual EBI_CSn line. When accessing one of the memory regions, the corresponding CSn line is asserted. This way up to 4 separate devices can share the EBI lines and be identified by the EBI_CSn line. Each bank can individually be enabled or disabled in the EBI_CTRL register.

The bank separation depends on whether the access originates from code space or not and on the setting of the ALTMAP bit in the EBI_CTRL register. From code space three 32 MB banks and one 128 MB bank can be accessed. From data space either four 64 MB banks (when ALTMAP bit is 0) or four 256 MB banks (when the ALTMAP bit is 1) can be accessed as shown in Figure 36.23 EBI Default Memory Map (ALTMAP = 0) on page 1317 and Figure 36.24 EBI Alternative Memory Map (ALTMAP = 1) on page 1318 respectively.

The EBI regions starting at address 0x80000000 in the memory map of the EFM32 Giant Gecko 12 can also be used for code execution. When running code via EBI regions starting at this address, the Cortex-M4 uses the System bus interface to fetch instructions. This results in reduced performance as the Cortex-M4 accesses stack, other data in SRAM and peripherals using the System bus interface. Code accesses via the System bus interface will not be cached. Furthermore, it should be noted that the address area from 0xA0000000 to 0xC0000000 is marked NX (no-execute) by default. To be able to run code via the EBI efficiently, the EBI is also mapped in the code space at address 0x12000000. When running code from this space, the Cortex-M4 fetches instructions through the I/D-Code bus interface, leaving the System bus interface for data access. Instructions fetched via the I/D-Code bus interface can be cached to increase performance. The EBI regions mapped into the code space can however only be accessed by the CPU, i.e. not the DMA.

Depending on the setting of the ITS bitfield in the EBI_CTRL register. The external device behavior, including for example data width, timing definitions, page mode operation, and pin polarities, is either defined for all banks at once or individually per bank.

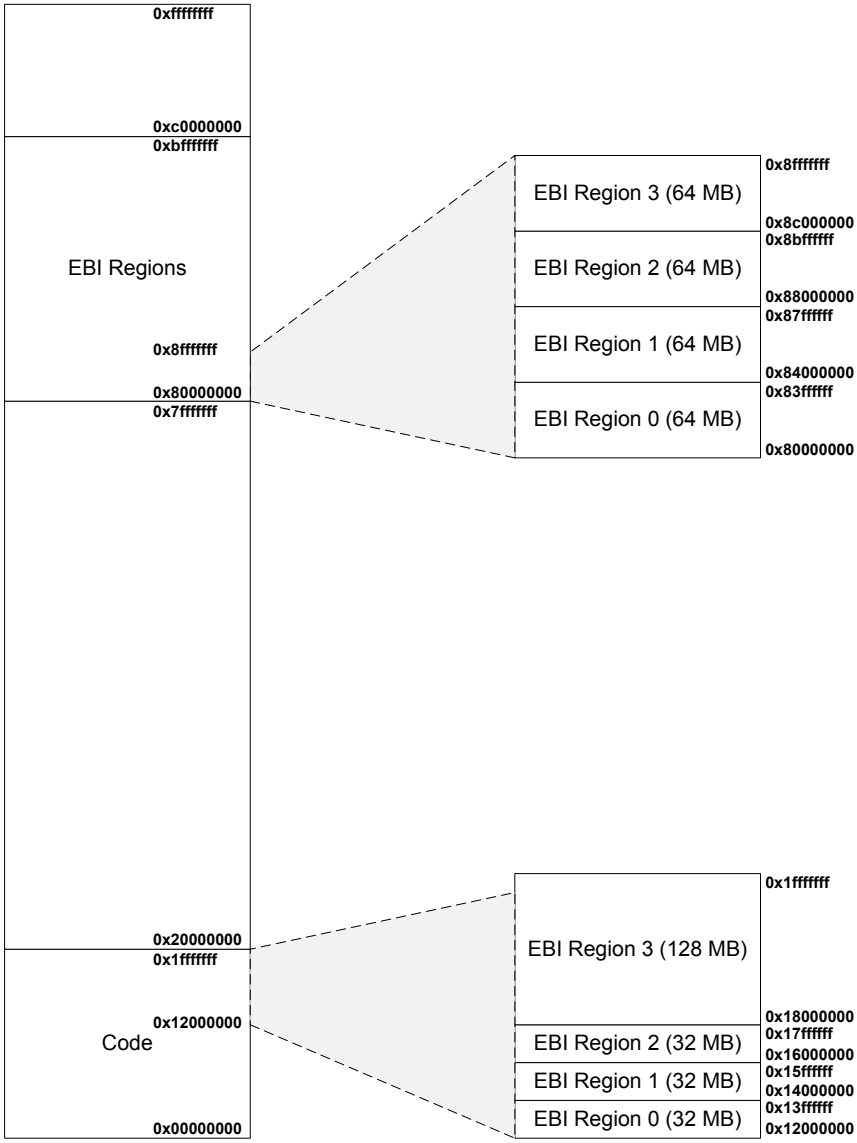


Figure 36.23. EBI Default Memory Map (ALTMAP = 0)

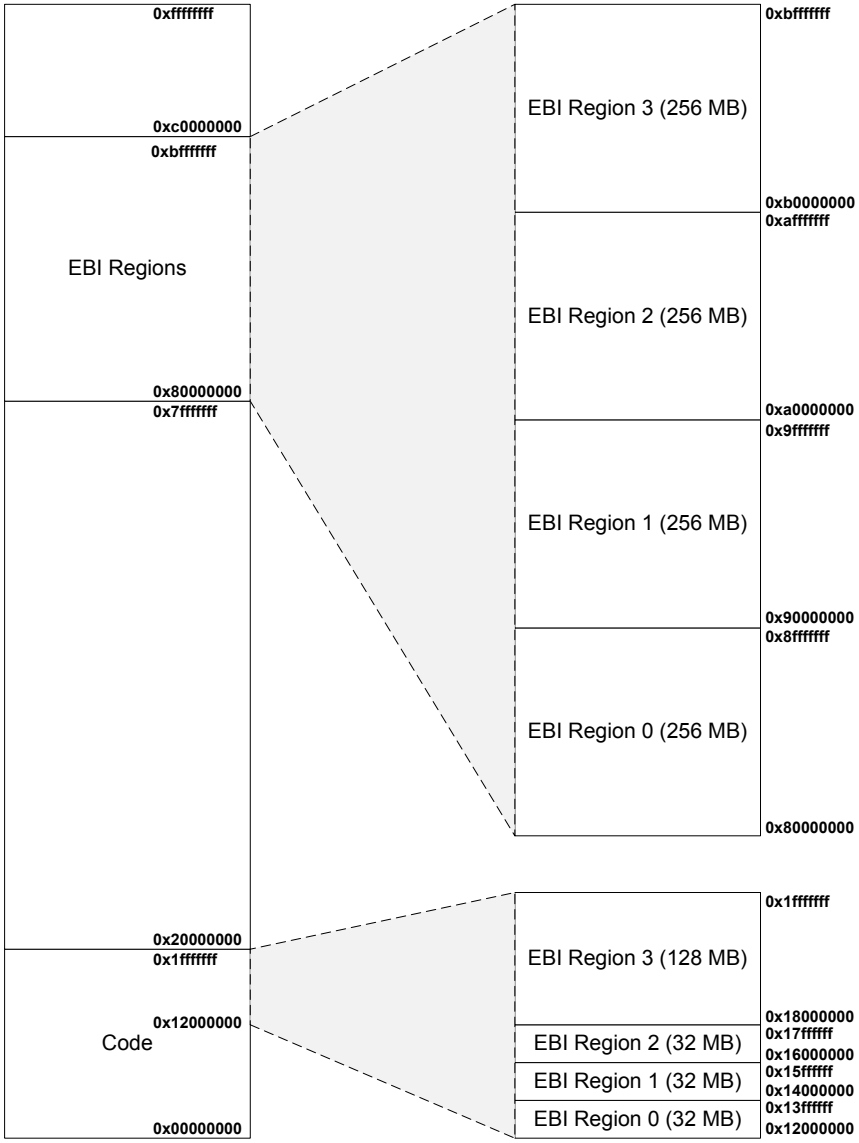


Figure 36.24. EBI Alternative Memory Map (ALTMAP = 1)

36.3.13 WAIT/ARDY.

Some external devices are able to indicate that they are not finished with either write or read operation by asserting the WAIT / ARDY line. This input signal is used to extend the REn/WEn cycles for slow devices. The interpretation of the polarity of this signal can be configured with the ARDYPOL bit in EBI_POLARITY. E.g. if the ARDYPOL is set to ACTIVELOW, then the REn/WEn cycle is extended while the ARDY line is kept low. The ARDY functionality is enabled by setting the ARDYEN bit in the EBI_CTRL register. It is also possible to enable a timeout check, which generates a bus error if the ARDY is not deasserted within the timeout period. This prevents a system lock up condition in the case that the external device does not deassert ARDY. The timeout functionality is disabled by setting ARDYTODIS in the EBI_CTRL register.

When the ITS bitfield in the EBI_CTRL register is set to 0, the wait behavior defined in the ARDYEN and ARDYTODIS bitfields applies to all 4 memory banks. When ITS is set to 1 each memory bank uses an individual wait behavior definition. In this case bitfields ARDY-EN and ARDYTODIS only apply to bank 0. Wait behavior for bank n is then defined in the ARDYnEN and ARDYTOnDIS bitfields.

36.3.14 NAND Flash Support

NAND Flash devices offer high density at relatively low cost when compared to NOR Flash devices. Unlike NOR Flash, which offers random read access, NAND Flash devices are based on page access and use an indirect interface. Furthermore, a NAND Flash can contain invalid bits leading to invalid blocks, which leads to requirements such as bit error detection/correction and bad block management.

The EBI offers support for glueless connection of a NAND Flash by implementing dedicated EBI_NANDREn and EBI_NANDWEn pins and by providing hardware for single error correction double error detection (SEC-DED) Error Correction Code (ECC) generation. NAND Flash support is enabled by setting the EN bitfield in the EBI_NANDCTRL register to 1. The BANKSEL bitfield in EBI_NANDCTRL defines which memory bank has a NAND Flash devices attached to it. NAND Flash data width, read timing, and write timing are programmed via the standard EBI registers as described in 36.3.14.2 Width and Timing Configuration. ECC support is described in 36.3.15 Error Correction Code.

Both standard and Chip Enable Don't Care (CEDC) NAND Flash devices are supported and they can be attached as shown in Figure 36.25 EBI Connection with Standard NAND Flash on page 1319 and Figure 36.26 EBI Connection with Chip Enable Don't Care NAND Flash on page 1319 respectively. For standard NAND Flash devices, the Chip Enable (CEn) pin needs to remain asserted low during the entire read cycle busy period, in which data is transferred from the memory array into the NAND Flash internal data registers in order to prevent an early return to standby mode. CEDC NAND Flash devices do not have this restriction, but they do not support the automatic sequential read function. For CEDC NAND Flash the shared EBI_REn and EBI_WEn pins can be used instead of the dedicated EBI_NANDREn and EBI_NANDWEn pins.

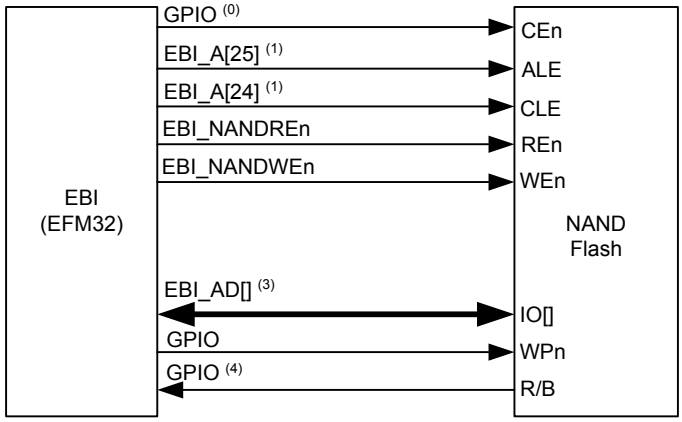


Figure 36.25. EBI Connection with Standard NAND Flash

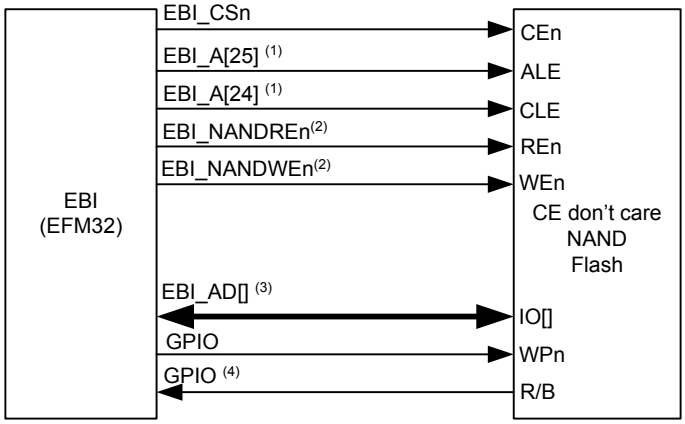


Figure 36.26. EBI Connection with Chip Enable Don't Care NAND Flash

Note:

- (0) For a standard NAND Flash the EBI_CSn should be left unconnected.

- (1) The address lines mapping to the NAND Flash ALE and CLE signals can be chosen as explained in 36.3.14.1 Register Selection
- (2) For a CEDC NAND Flash the shared EBI_REn and EBI_WEn pins can be used instead of the dedicated EBI_NANDREn and EBI_NANDWEn pins
- (3) Both 8-bit and 16-bit NAND Flash are supported.
- (4) The NAND Flash ready/busy (R/B) signal should be observed via GPIO (not via EBI_ARDY)

36.3.14.1 Register Selection

NAND Flash uses an indirect I/O interface in which the NAND Flash is controlled by programming the NAND Flash internal Command, Address, and Data registers. NAND Flash does not use dedicated address lines. Because of this indirect I/O interface the NAND Flash memory size is not restricted by the memory map of the EFM32 Giant Gecko 12. The NAND Command, Address, and Data registers can be accessed via memory mapped IO in which two address lines are chosen for connection with the ALE and CLE signals. The memory mapping and the two used address lines should be chosen such that they adhere to the ALE/CLE encoding shown in Table 36.4 EBI NAND Flash Register Select on page 1320. Either EBI_A or EBI_AD address lines can be used as long as the chosen addressing mode does not multiplex data signals onto the chosen lines. The EBI_A[25:24] address lines used in Figure 36.25 EBI Connection with Standard NAND Flash on page 1319 and Figure 36.26 EBI Connection with Chip Enable Don't Care NAND Flash on page 1319 are just an example.

Table 36.4. EBI NAND Flash Register Select

ALE	CLE	Selected NAND Flash Register
0	0	Data Register
0	1	Command Register
1	0	Address Register
1	1	Undefined

36.3.14.2 Width and Timing Configuration

The regular EBI registers are used for defining transfer width, read timing, and write timing for the transactions on the NAND Flash interface. NAND Flash specific parameters as for example block size or the number of address cycles are not configured in the EBI and need to be dealt with via driver software. Also higher level tasks as for example wear-leveling, bad block management, and logical-to-physical block mapping should be addressed via driver software.

External transaction width is defined via the address mode as defined in MODE field of EBI_CTRL. As only 3 NAND Flash registers are memory mapped it suffices to use either the D8A8 or D16 address mode. The D16A16ALE and D8A24ALE address modes can also be used, but they require unnecessary external address latch cycles and/or circuitry. For a 8-bit wide NAND Flash device, the D8A8 address mode is therefore recommended, whereas for a 16-bit wide NAND Flash device the D16 address mode is recommended. If the AHB transaction width does not match the external NAND device transaction width, then automatic transaction translation is performed as described in 36.3.11 Data Access Width. Note that a bus fault is generated in case of an 8-bit write to a 16-bit NAND device as neither byte lanes nor read-modify-write is supported for NAND Flash.

NAND Flash write timing is defined in the EBI_WRTIMING(n) register. Figure 36.27 EBI NAND Flash Command Latch Timing on page 1321, Figure 36.28 EBI NAND Flash Address Latch Timing on page 1322, and Figure 36.29 EBI NAND Flash Data Input Timing on page 1322 show the command latch, address latch and data input timing respectively assuming the D8A8 address mode with EBI_AD[x] used as ALE and EBI_AD[y] used as CLE.

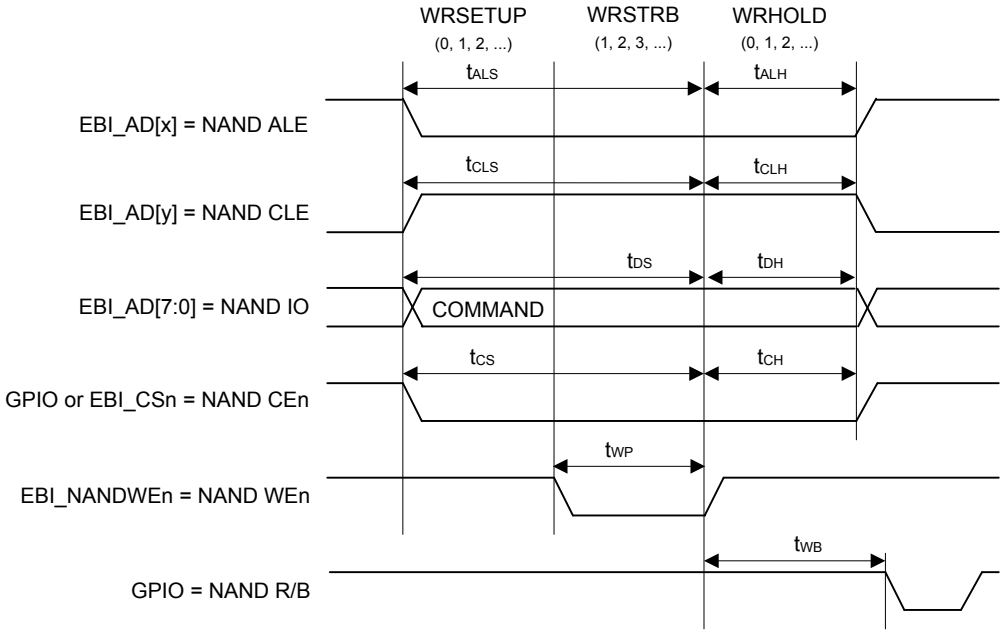


Figure 36.27. EBI NAND Flash Command Latch Timing

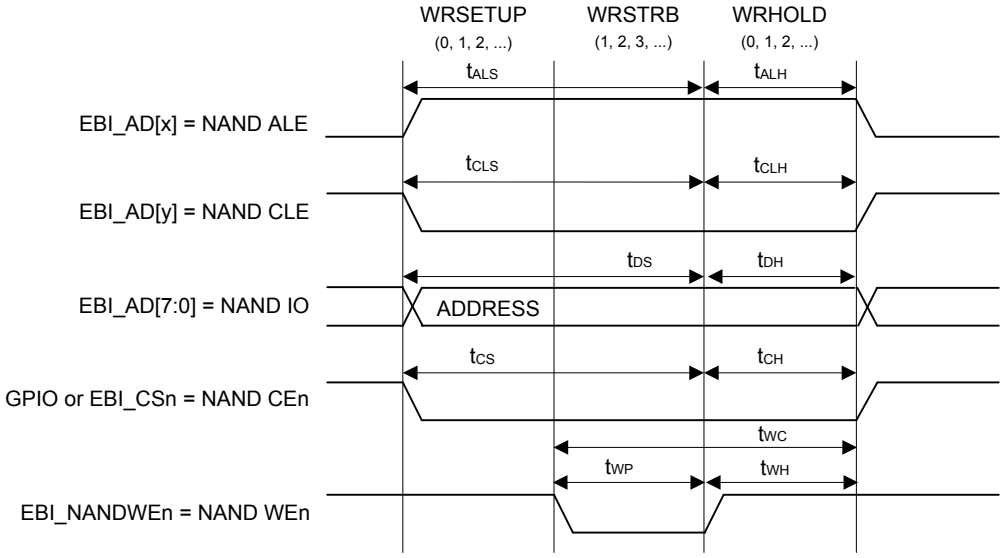


Figure 36.28. EBI NAND Flash Address Latch Timing

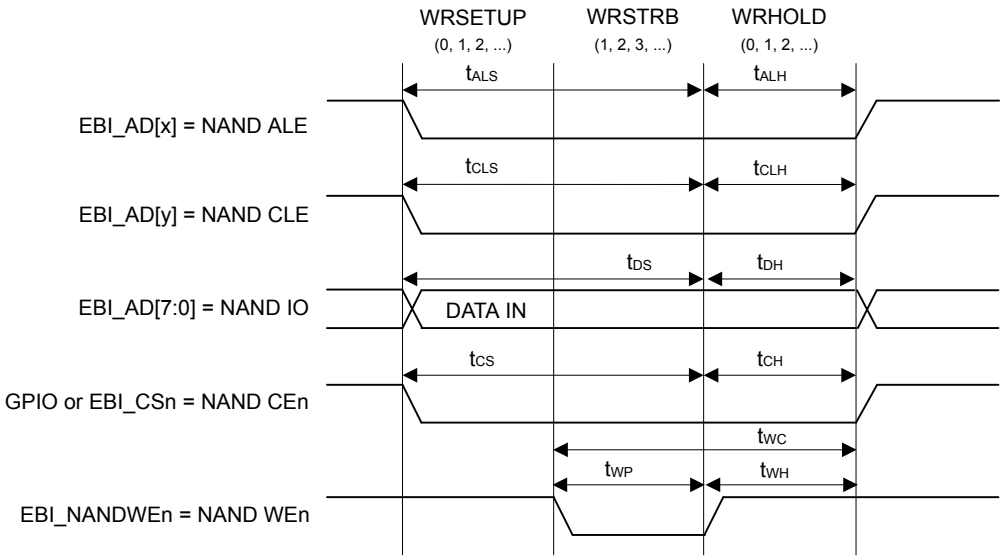


Figure 36.29. EBI NAND Flash Data Input Timing

The EBI_WRTIMING(n) setting requirements for satisfying the NAND Flash timing parameters for command latching, address latching and data input timing are shown in Table 36.5 EBI NAND Flash Write Timing on page 1322.

Table 36.5. EBI NAND Flash Write Timing

NAND Flash Write Timing Parameter	EBI Write Timing Parameter Requirements
tADL	<= t(WRHOLD) + t(WRSETUP) + t(WRSTRB)
tALS	<= t(WRSETUP) + t(WRSTRB)
tCS	<= t(WRSETUP) + t(WRSTRB)
tCLS	<= t(WRSETUP) + t(WRSTRB)

NAND Flash Write Timing Parameter	EBI Write Timing Parameter Requirements
tDS	<= t(WRSETUP) + t(WRSTRB)
tALH	<= t(WRHOLD)
tCH	<= t(WRHOLD)
tCLH	<= t(WRHOLD)
tDH	<= t(WRHOLD)
tWC	<= t(WRHOLD) + t(WRSETUP) + t(WRSTRB)
tWH	<= t(WRHOLD) + t(WRSETUP)
tWP	<= t(WRSTRB)
tWB	(R/B edges can be detected by edge triggered GPIO interrupts)

NAND Flash read timing is defined in the EBI_RDTIMING(n) register. Figure 36.30 EBI NAND Flash Data Output Timing on page 1323 shows the NAND Flash data output timing assuming the D8A8 address mode.

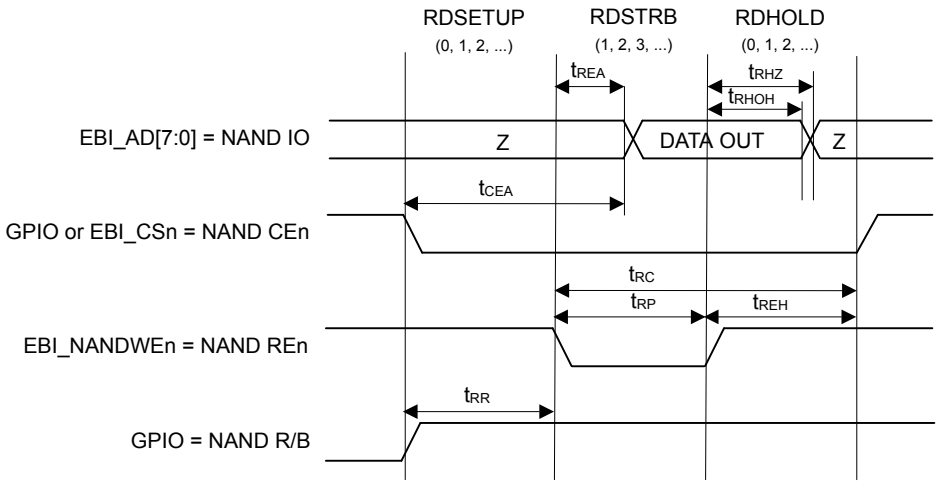


Figure 36.30. EBI NAND Flash Data Output Timing

The EBI_RDTIMING(n) setting requirements for satisfying the NAND Flash timing parameters for data output timing are shown in Table 36.6 EBI NAND Flash Read Timing on page 1323.

Table 36.6. EBI NAND Flash Read Timing

NAND Read Timing Parameter	EBI Read Timing Parameter Requirements
tCEA	<= t(RDSETUP) + t(RDSTRB)
tREA	<= t(RDSTRB)
tRP	<= t(RDSTRB)
tRHZ	<= t(RDHOLD)
tREH	<= t(RDHOLD) + t(RDSETUP)
tRC	<= t(RDHOLD) + t(RDSETUP) + t(RDSTRB)
tRR	<= t(RDSETUP) (assuming software wait for R/B high)
tAR	<= t(RDSETUP)

NAND Read Timing Parameter	EBI Read Timing Parameter Requirements
tCLR	<= t(RDSETUP)
tIR	<= t(RDSETUP)

The NAND Flash timing parameters tWHR and tRHW define separation of read and write pulses and therefore they can be satisfied by a combination of EBI_RDTIMING(n) and EBI_WRTIMING(n) settings as shown in Table 36.7 EBI NAND Flash Read/Write Timing Requirements on page 1324.

Table 36.7. EBI NAND Flash Read/Write Timing Requirements

NAND Timing Parameter	EBI Timing Parameter
tWHR	<= t(WRHOLD) + t(RDSETUP)
tRHW	<= t(RDHOLD) + t(WRSETUP)

Remaining NAND Flash timing parameters, e.g. tRST and tPROG, should be dealt with in software.

36.3.14.3 Application Examples

A typical 528-byte page read sequence for an 8-bit wide NAND Flash is as follows:

- Configuration: Enable and select the memory bank connected to the NAND Flash device via the EN and BANKSEL bitfields in the EBI_NANDCTRL register. Set the MODE field of the EBI_CTRL register to D8A8 indicating that the attached device is 8-bit wide. Program the EBI_RDTIMING and EBI_WRTIMING registers to fulfill the NAND timing requirements.
- Command and address phase: Program the NAND Command register to the page read command and program the NAND Address register to the required read address. This can be done via Cortex-M4 or DMA writes to the memory mapped NAND Command and Address registers. The automatic data access width conversions described in 36.3.11 Data Access Width can be used if desired to for example automatically perform 4 consecutive address byte transactions in response to one 32-bit word AHB write to the NAND Address register (in this case the 2 address LSBs should not be used to map onto the NAND ALE/CLE signals).
- Data transfer phase: Wait for the NAND Flash internal data transfer phase to complete as indicated via its ready/busy (R/B) pin. The user can use the GPIO interrupt functionality for this. The 528-byte data is now ready for sequential transfer from the NAND Flash Data register.
- Read phase: Clear the ECC_PARITY register and start Error Code Correction (ECC) parity generation by setting both the ECCSTART and ECCCLEAR bitfields in the EBI_CMD register to 1. Now all subsequently transferred data to/from the NAND Flash devices is used to generate the ECC parity code into the EBI_ECCPARITY register. Read 512 subsequent bytes of main area data from the NAND Flash Data register via DMA transfers. This can for example be done via 32-bit word DMA transfers (as long as the two address LSBs are not used to map onto the NAND ALE/CLE signals). Stop ECC parity generation by setting the ECCSTOP bitfield in the EBI_CMD register to 1 so that following transactions will not modify the parity result. Read out the final 16 bytes from the NAND Flash spare data area.
- Error correction phase: Compare the ECC code contained in the read spare area data against the computed ECC code from the EBI_ECCPARITY register. The user software can accept, correct, or discard the read data according the comparison result. No automatic correction is performed.

A typical 528-byte page program sequence for an 8-bit wide NAND Flash is as follows:

- Configuration: Configure the EBI for NAND Flash support via the EBI_NANDCTRL, EBI_CTRL, EBI_RDTIMING and EBI_WRTIMING registers.
- Command and address phase: Program the NAND Command register to command for page programming (serial data input) and program the NAND Address register to the desired write address.
- Write phase: Clear the ECC_PARITY register and start Error Code Correction (ECC) parity generation by setting both the ECCSTART and ECCCLEAR bitfields in the EBI_CMD register to 1. Now all subsequently transferred data to/from the NAND Flash devices is used to generate the ECC parity code into the EBI_ECCPARITY register. Write 512 subsequent bytes of user main data to the NAND Flash Data register via for example DMA transfers. Stop ECC parity generation and read out the computed ECC parity data from EBI_ECCPARITY. Write the final 16 bytes of spare data including the computed ECC parity data bytes.
- Program phase: Write the auto program command to the NAND Flash Command register after which the NAND Flash will indicate that it is busy via its read/busy (R/B) pin. After read/busy goes high again, the success of the program command can be verified by programming the read status command.

36.3.15 Error Correction Code

The EBI provides provides hardware support for generation of an Error Correction Code (ECC). The used ECC is a Hamming (Hsiao) code providing single bit error correction and double error detection (SEC-DED). ECC can be used to detect and/or correct failing bits in a NAND Flash page. ECC generation is enabled by setting bitfield ECCSTART in the EBI_CMD register to 1. All subsequent data traffic to/from the memory bank specified in the BANKSEL bitfield of the EBI_NANDCTRL register is then used for generation of the ECC into the EBI_ECCPARITY register independent of the address in that bank. ECC generation is stopped by writing 1 to the ECCSTOP bitfield in the EBI_CMD register. The EBI_ECCPARITY register is cleared by writing 1 to the ECCCLEAR register. The ECCACT status bit in the EBI_STATUS register shows whether ECC generation is active or not.

The ECC computation is as shown in Figure 36.31 EBI ECC Generation on page 1325 and Table 36.8 EBI ECC Bit/Column Parity on page 1325. Although the table only shows the ECC generation for 8-bit data transfers, the ECC hardware also works for 16-bit data transfers. In that case only the interpretation of the parity bits is different.

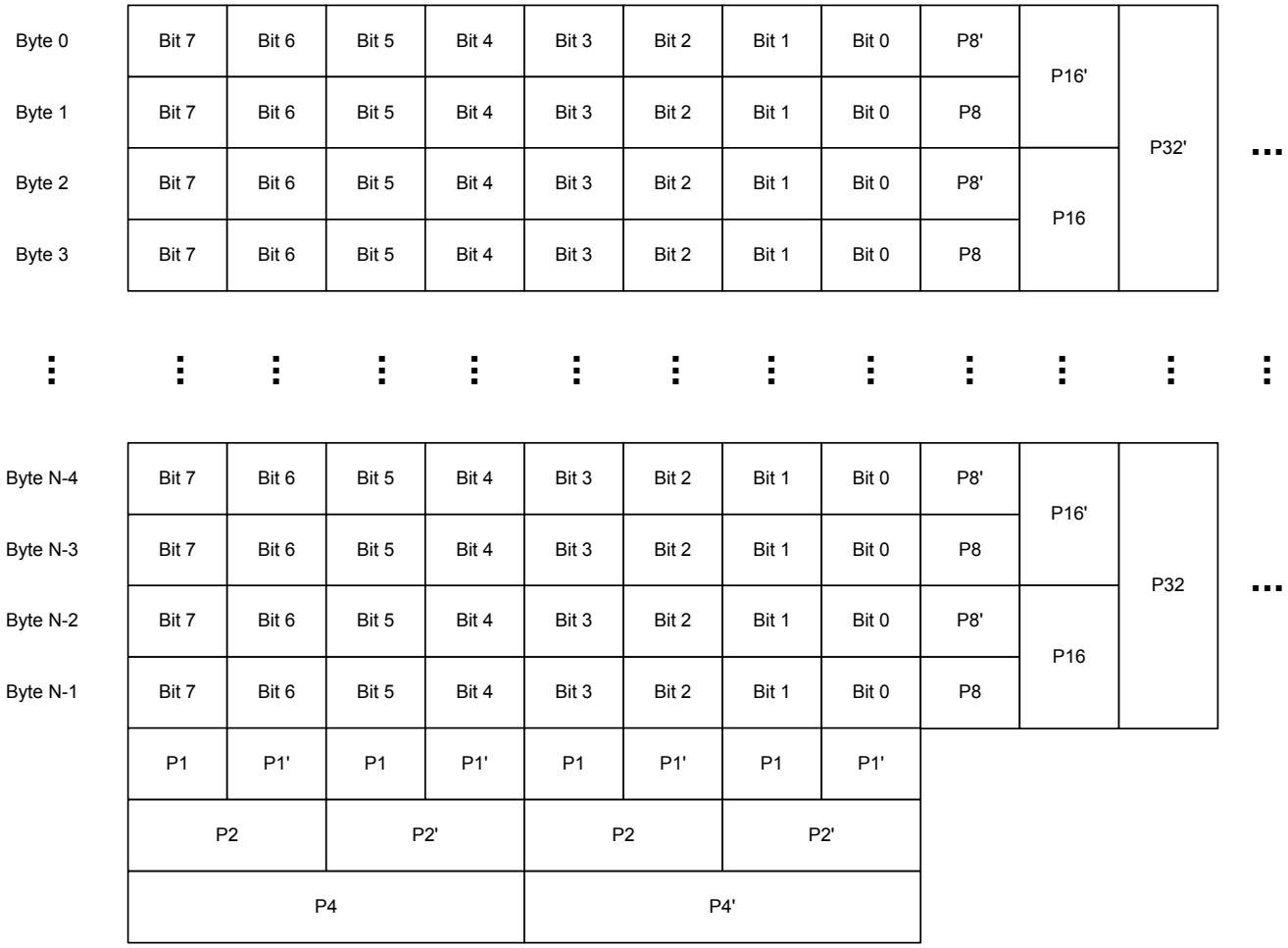


Figure 36.31. EBI ECC Generation

Table 36.8. EBI ECC Bit/Column Parity

Parity bit	Generation for 8-bit data
P1'	Bit 6 xor Bit 4 xor Bit 2 xor Bit 0 xor P1'
P1	Bit 7 xor Bit 5 xor Bit 3 xor Bit 1 xor P1
P2'	Bit 5 xor Bit 4 xor Bit 1 xor Bit 0 xor P2'
P2	Bit 7 xor Bit 6 xor Bit 3 xor Bit 2 xor P2
P4'	Bit 3 xor Bit 2 xor Bit 1 xor Bit 0 xor P4'

Parity bit	Generation for 8-bit data
P4	Bit 7 xor Bit 6 xor Bit 5 xor Bit 4 xor P4

Table 36.9. EBI ECC Byte/Row Parity

Parity bit	Generation for 8-bit data
RP(x)	Byte(x)(7) xor Byte(x)(6) xor Byte(x)(5) xor Byte(x)(4) xor Byte(x)(3) xor Byte(x)(2) xor Byte(x)(1) xor Byte(x)(0)
P8'	RP(0) xor RP(2) xor RP(4) xor RP(6) xor ... xor RP(N-4) xor RP(N-2)
P8	RP(1) xor RP(3) xor RP(5) xor RP(7) xor ... xor RP(N-3)xor RP(N-1)
P16'	RP(0) xor RP(1) xor RP(4) xor RP(5) xor ... xor RP(N-4) xor RP(N-3)
P16	RP(2) xor RP(3) xor RP(6) xor RP(7) xor ... xor RP(N-2) xor RP(N-1)
Etc.	Etc.

The generated ECC code can be read from the EBI_ECCPARITY register according to the format shown in Figure 36.32 EBI EBI_ECCPARITY Format on page 1326. The number of valid ECC bits depends on the number of transferred bytes during the time that the ECC hardware is running as indicated in Table 36.10 EBI EBI_ECCPARITY valid bits on page 1326.

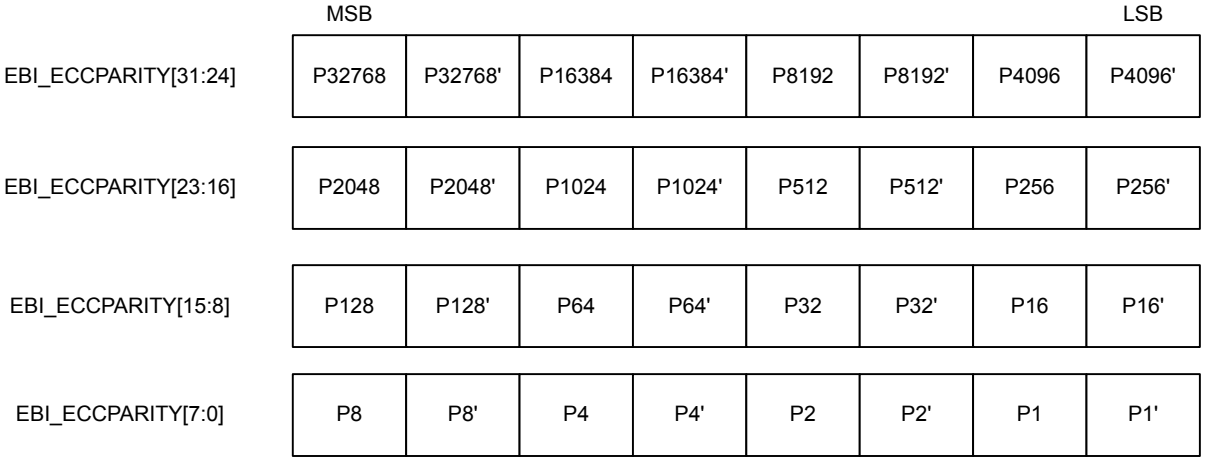


Figure 36.32. EBI EBI_ECCPARITY Format

Table 36.10. EBI EBI_ECCPARITY valid bits

Number of data bytes used for ECC generation	Valid EBI_ECCPARITY bits
256	EBI_ECCPARITY[21:0]
512	EBI_ECCPARITY[23:0]
1024	EBI_ECCPARITY[25:0]

Number of data bytes used for ECC generation	Valid EBI_ECCPARITY bits
2048	EBI_ECCPARITY[27:0]
4096	EBI_ECCPARITY[29:0]
8192	EBI_ECCPARITY[31:0]

Software can compare, XOR, the parity data generated in EBI_ECCPARITY with the parity information stored in the spare area for the used data set. The syndrome resulting from XOR'ing the valid EBI_ECCPARITY bits with the ECC code read from the spare area can be used for error detection and correction as shown in Table 36.11 EBI Error Detection Result on page 1327.

Table 36.11. EBI Error Detection Result

Error Detection Result	Syndrome	Interpretation
No Error	Syndrome has all valid Pn, Pn' bits 0	No error has been detected
1-bit Correctable Error	For all valid syndrome (Pn, Pn') pairs: Pn = not(Pn')	1 bit in the user main data is incorrect and it can be corrected. For 8-bit wide data the position of the incorrect bit is indicated by bit pattern (P4, P2, P1); the position of the incorrect byte is indicated by (..., P32, P16, P8). For 16-bit wide data the position of the incorrect bit is (P8, P4, P2, P1); the incorrect byte number is indicated by (..., P64, P32, P16)
ECC Error	1 bit of the XOR result is high	An error has been detected in the ECC itself. No error has been detected in the user data
Uncorrectable Error	Other cases	Multiple (2 or more) bits are incorrect. This error cannot be corrected

36.3.16 TFT Direct Drive

TFT Direct Drive can be used to automatically transfer frame data stored in either internal or external memory to a TFT display without frame buffer. The EBI generates the necessary RGB control signals for the TFT display and it coordinates and aligns the pixel data transfers accordingly. The Direct Drive engine is enabled by setting the DD bitfield in the EBI_TFTCTRL register to either INTERNAL or EXTERNAL. The RGB interface consists of 8 or 16 data lines on EBI_AD together with the EBI_DATAEN, EBI_VSYNC, EBI_HSYNC and EBI_DCLK control signals. EBI_TFTCSn indicates whether the DD bitfield is programmed to DISABLED or not. Whether Direct Drive is active or not can also be read via the DDACT status bit in the EBI_STATUS register.

The dimensions of the visible display are defined in the VSZ and HSZ bitfields of the EBI_TFTSIZE register. Hardware automatically adds 1 to the size programmed in these bitfields. The front and back porch sizes are defined in the HFPORCH, HBPORCH, VFPORCH and VBPORCH bitfields of the EBI_TFTHPORCH and EBI_TFTVPORCH registers. The porch and visible display sizes define the number of EBI_DCLK pulses per line and the number of lines per frame according to Figure 36.33 EBI TFT Total Width on page 1328 and Figure 36.34 EBI TFT Total Height on page 1328 respectively.

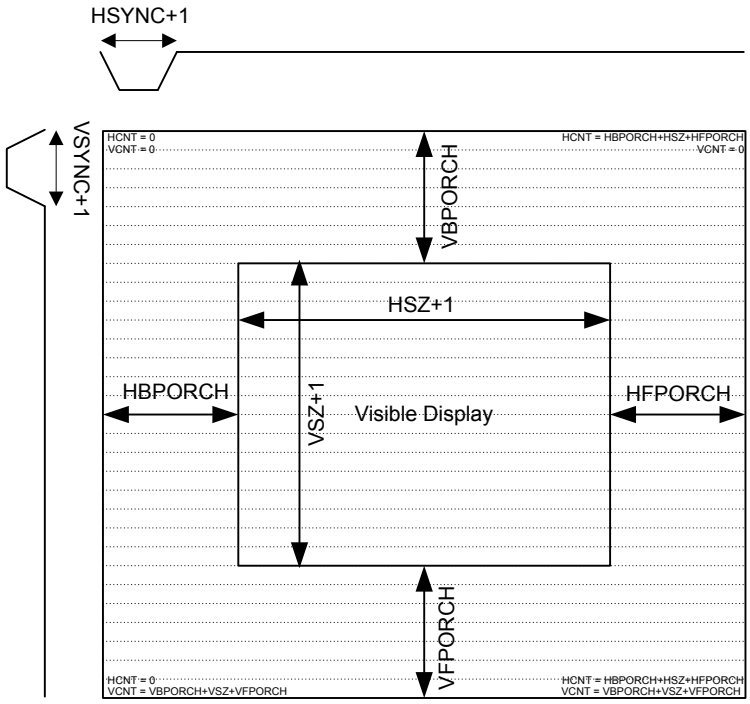
Number of EBI_DCLK pulses per line = HBPORCH + (HSZ + 1) + HFPORCH

Figure 36.33. EBI TFT Total Width

Number of lines per frame = VBPORCH + (VSZ + 1) + VFPORCH

Figure 36.34. EBI TFT Total Height

The horizontal and vertical synchronization pulses begin at the starts of the horizontal and vertical back porch intervals respectively. For the HSYNC pulse a delayed start position can be defined in the HSYNCSTART bitfield of the EBI_TFTHPORCH register. The end of the HSYNC pulse is not delayed and therefore the HSYNC pulse width is shortened when using a non-zero HSYNCSTART. The widths, or rather end positions, of the HSYNC and VSYNC synchronization pulses are defined in the HSYNC and VSYNC bitfields of the EBI_TFTSIZE register respectively. The horizontal synchronization pulse width is specified in pixels. The vertical synchronization pulse width is specified in lines. Hardware automatically adds 1 to the width programmed in these bitfields. The EBI_TFTSIZE bitfields are shown in Figure 36.35 EBI TFT Size on page 1329. When Direct Drive is enabled, the VCNT and HCNT bitfields in the EBI_TFTSTATUS register show how the frame display progresses. VCNT is a counter containing the current line position in a frame. It counts from 0 (first line in the vertical back porch) to VBPORCH + VSZ + VFPORCH (last line in the vertical front porch). HCNT is a counter containing the current pixel position within a line. It counts from 0 (first pixel in the horizonal back porch) to HBPORCH + HSZ + HFPORCH (last pixel in the horizontal front porch).



Total width = HBPORCH + (HSZ + 1) + HFPORCH

Total height = VBPORCH + (VSZ + 1) + VFPORCH

Figure 36.35. EBI TFT Size

While the Direct Drive engine is transferring frame data from internal or external memory to the TFT, the EBI can still be used for other EBI transfers to external devices. The interleaving of such EBI transfers with transfers originating from the Direct Drive engine is controlled via the INTERLEAVE field in the EBI_TFTCTRL register. Interleaving can be limited to occur only during the vertical and horizontal porch intervals by setting the INTERLEAVE field to PORCH. EBI accesses outside the porch intervals while INTERLEAVE is set to PORCH can cause the insertion of a high number of wait states on the AHB bus. In case the TFT dot clock EBI_DCLK is relatively slow compared to the external device access time, interleaving can also be allowed during the active interval of the TFT by setting the INTERLEAVE bitfield to ONEPERDCLK or UNLIMITED. In both cases interleaving during the porch intervals is unlimited as it is when the PORCH setting is used. If INTERLEAVE is set to ONEPERDCLK then at most 1 EBI access is inserted per EBI_DCLK period in the active display interval at the point immediately after the pixel transfer. Wait states are inserted on the AHB bus while waiting for this insertion point. The access time of such an interleaved transfer should be guaranteed by software to fit in the free interval between pixel transfers as indicated in Figure 36.47 EBI TFT Pixel Timing on page 1336. If INTERLEAVE is set to UNLIMITED, which is the default, then there are no restrictions on performing EBI transactions during Direct Drive operation. Although transactions related to Direct Drive have priority over other EBI transactions, jitter on the EBI_DCLK can be introduced in case an EBI transaction is ongoing while the Direct Drive engine wants to insert its next transaction. In case the programmed EBI_DCLK period can not be met, the DDJIT interrupt flag in the EBI_IF register is set and the EBI_DCLK period is stretched to accommodate the delayed pixel data.

Note: If INTERLEAVE is limited to PORCH only and zero porch sizes are programmed in the EBI_TFTHPORCH and EBI_TFTVPORCH registers, then no slots are left open for interleaving traffic and therefore interleaving EBI accesses can never finish.

36.3.16.1 Direct Drive From Internal Memory

Any internal memory can be used as the frame source location for Direct Drive. Direct Drive display from internal memory is started by setting the DD bitfield in the EBI_TFTCTRL register to INTERNAL. The TFT controller indicates that the pixel buffer EBI_TFTDD is empty and needs to be filled by raising the corresponding DMA request. This DMA request is initially set and it is cleared when EBI_TFTDD is written. It is set again once the pixel data has been transferred to the display. One DMA request is generated for each visible pixel. The Direct Drive engine will automatically align the data written to EBI_TFTDD according to the setup and hold requirements with respect to EBI_DCLK and send it out to the TFT via the EBI_AD lines. Whether the EBI_TFTDD buffer is full or empty is also signaled by the DDEMPTY interrupt flag in the EBI_IF register and by the TFTDDEMPTY status bit in the EBI_STATUS register. Given the relatively low performance of using software polling and interrupts compared to using DMA, these non-DMA mechanisms are only advised for very low pixel rates. If pixel data is not provided in time the EBI_DCLK will be stretched to accommodate the late pixel data and the Direct Drive Jitter interrupt flag DDJIT in the EBI_IF register is set. Figure 36.36 EBI TFT Direct Drive from Internal Memory on page 1330 shows the setup for Direct Drive from internal memory.

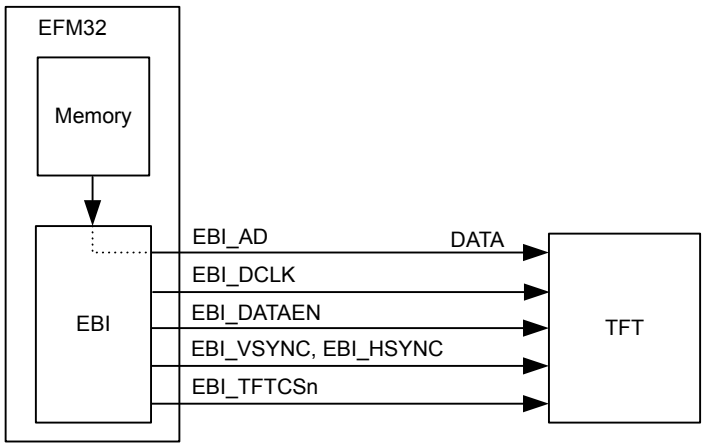


Figure 36.36. EBI TFT Direct Drive from Internal Memory

36.3.16.2 Direct Drive From External Memory

Direct Drive can also use an external memory bank as the frame source location. The used bank is defined in the BANKSEL bitfield of the EBI_TFTCTRL register. Direct Drive display from external memory is started by setting the DD bitfield in the EBI_TFTCTRL register to EXTERNAL. Data is then streamed directly from the external memory to the TFT. Figure 36.37 EBI TFT Direct Drive from External Memory (non-multiplexed address/data) on page 1331 and Figure 36.38 EBI TFT Direct Drive from External Memory (multiplexed address/data) on page 1331 show the setup for Direct Drive from external memory when using non-multiplexed and multiplexed address and data lines respectively.

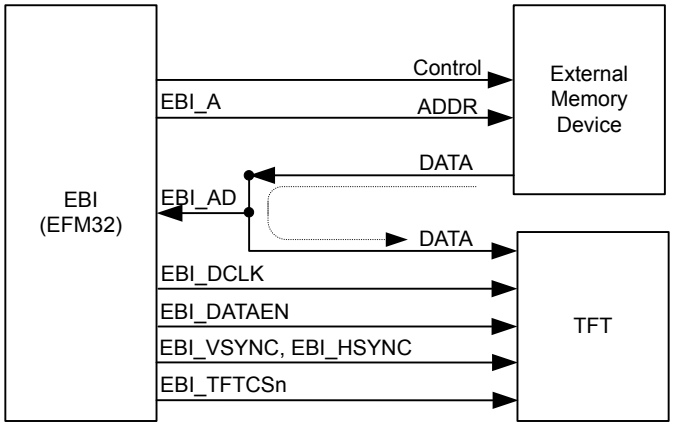


Figure 36.37. EBI TFT Direct Drive from External Memory (non-multiplexed address/data)

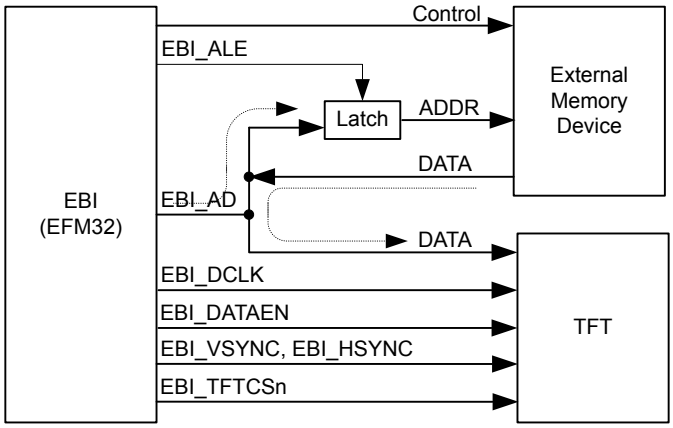
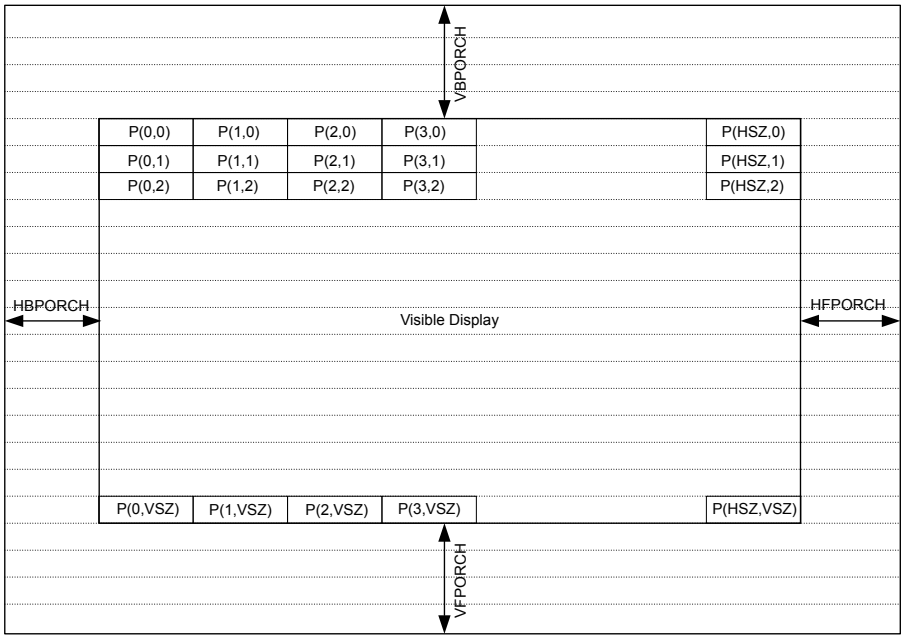


Figure 36.38. EBI TFT Direct Drive from External Memory (multiplexed address/data)

The start address for the frame transfer is defined in the EBI_TFTFRAMEBASE register. The Direct Drive address is automatically incremented for each visible pixel and it does therefore not depend on the programmed porch sizes. The address increment depends on the WIDTH bitfield in the EBI_TFTCTRL register. The increment per visible pixel is 1 if the WIDTH bitfield in the EBI_TFTCTRL register is programmed to BYTE and it is 2 if WIDTH is programmed to HALFWORD. Additionally a horizontal stride is added to the Direct Drive address at the end of each visible line. This stride can be programmed in the HSTRIDE bitfield of the EBI_TFTSTRIDE register. The first visible pixel always corresponds to the address defined in the EBI_TFTFRAMEBASE register. On either the vertical or horizontal synchronization event, as defined in the FBCTRIG bitfield of the EBI_TFTCTRL register, the EBI_TFTFRAMEBASE register is copied into an internal frame base buffer (FBC). This allows software to reprogram the EBI_TFTFRAMEBASE register based on VSYNC or HSYNC interrupts, which in turn can be used to for example implement double buffering or scrolling schemes. The HSYNC and VSYNC interrupts are generated at the same time as the local copy of EBI_TFTFRAMEBASE is made. If software reprograms EBI_TFTFRAMEBASE in the interrupt service routine, then the new value will only be used for address generation of the next line (in case FBCTRIG equals HSYNC) or the next frame (in case FBCTRIG equals VSYNC). For example, when FBCTRIG equals HSYNC and the interrupt service routine triggered by the HSYNC interrupt reads VCNT as 0, then a software update of EBI_TFTFRAMEBASE will take effect for Direct Drive addresses of the line which corresponds to a VCNT value of 1. Note that the EBI_TFTSTRIDE register is not relevant in

case the FBCTRIG is set to HSYNC as the HSYNC events reloads the internal frame base copy (FBC) with EBI_TFTFRAMEBASE at the start of each line. The Direct Drive address computation is summarized in Figure 36.39 EBI Direct Drive Address on page 1332.



FBCTRIG = VSYNC:

Local frame base copy FBC gets assigned with EBI_TFTFRAMEBASE on every EBI_VSYNC stobe.

Direct Drive Address for pixel P(x,y) = FBC + (x * PSZ) + (y * ((PSZ * (HSZ + 1)) + HSTRIDE))

FBCTRIG = HSYNC:

Local frame base copy FBC gets assigned with EBI_TFTFRAMEBASE on every EBI_HSYNC stobe.

Direct Drive Address for pixel P(x,y) = FBC + (x * PSZ)

The address increment per pixel (PSZ) is 1 if the WIDTH bitfield in EBI_TFTCTRL is programmed to BYTE and 2 if the WIDTH bitfield is programmed to HALFWORD.

Figure 36.39. EBI Direct Drive Address

Note: In case that the memory bank used for external Direct Drive is defined as 16-bit wide, then the Direct Drive address is internally shifted one bit to the right before being output on the EBI_AD or EBI_A lines.

The same bank defined with the BANKSEL field used for external Direct Drive may also be aliased for data access. This may be useful in systems with limited memory resources. Bank aliasing is enabled with the ALIASBANKEN bit in EBI_TFTCTRL. When EBI_TFTCTRL is set to 1, the bank selected by ALIASBANK will be aliased back into the bank selected by BANKSEL. Data access to the aliased memory region will target the designated Direct Drive bank instead. For example, if ALIASBANK is 1, BANKSEL is 0, and ALIASBANKEN is set to 1, any access to bank 1 will redirect to bank 0.

36.3.17 Alpha Blending and Masking

Automatic alpha blending and masking can be performed on AHB data written to or via the EBI. Alpha blending combines a foreground color with a background color into a new blended color and is further described in 36.3.17.1 Alpha Blending. Masking is a mechanism to suppress writes matching a specific color. It is used to preserve the background color and is further described in 36.3.17.2 Masking. Masking, if enabled, is applied before alpha blending as shown in Figure 36.40 EBI TFT Alpha Blending and Masking on page 1333. Masking and alpha blending can be used for both internal and external data transfers.

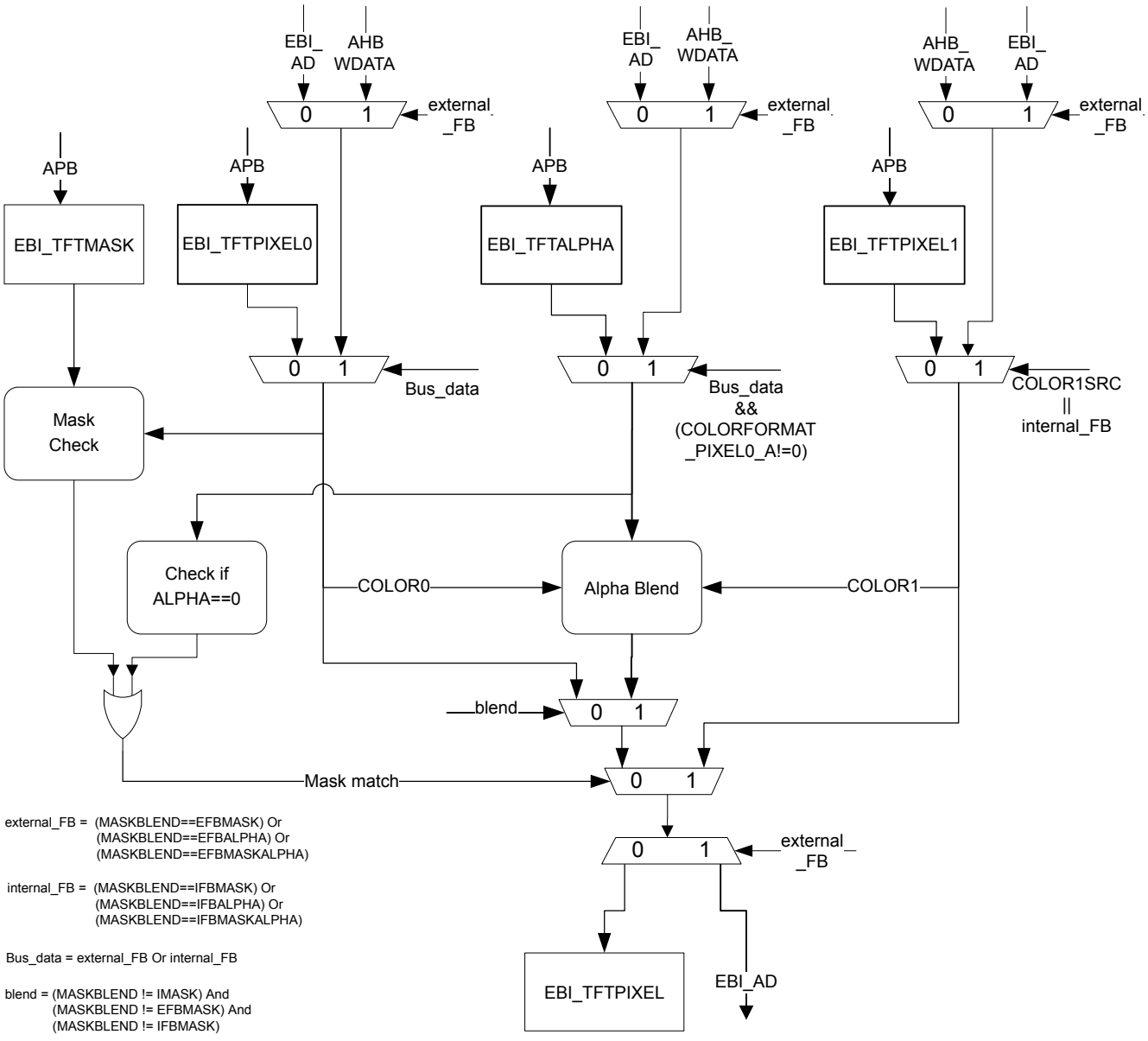


Figure 36.40. EBI TFT Alpha Blending and Masking

36.3.17.1 Alpha Blending

Automatic alpha blending can be performed on AHB data written to or via the EBI. Alpha blending can be enabled for either internal or external writes by setting the MASKBLEND bitfield in the EBI_TFTCTRL register. Internal writes are writes to the internal EBI_TFTPIXEL0 register. External writes are writes to the external device attached to the bank (Graphic Bank) defined in the BANKSEL bitfield of the EBI_TFTCTRL register. Alpha blending works on up to three operands, Color0, Color1 and Alpha/Mask. Foreground color, Color0 which can be in {R0, G0, B0} or {A0, R0, G0, B0} format. And a background color, Color1 in {R1, G1, B1}. The Color0 operand can be encoded in any of these ARGB formats: 0555, 0565, 0666, 0888, 5555, 6565, 6666 or 8888. The Color1 operand can be encoded in these RGB formats: 555, 565, 666, 888. Color0 ARGB format are as defined in the PIXEL0FORMAT bitfield of the EBI_TFTCOLORFORMAT register. Color1 RGB format are as defined in the PIXEL1FORMAT bitfield in the same register. The ALPHA operand use for the calculation can either come from the EBI_TFTALPHA register if this Color0 are 0555, 0565, 0666 or 0888. If Color0 is of the other 4 allowable formats, the ALPHA operand {A0} will be extracted from the Color0. In all cases except when 888 RGB format is used, Color0 and Color1 are promoted to RGB 888 before the alpha blending operation. The resultant Color always take on the format as defined in PIXEL1FORMAT bitfield. Alpha blending is performed according to formula Figure 36.41 EBI Alpha Blending Equation on page 1334. And Alpha value is select according to formula Figure 36.42 EBI Alpha Value Equation on page 1334.

AlphaBlend(Color0, Color1) = (({R0, G0, B0} x ALPHA) + ({R1, G1, B1} x (255 - ALPHA))) / 255

Figure 36.41. EBI Alpha Blending Equation

Alpha = TFTCOLORFORMAT_PIXEL0_A == 0 ? EBI_TFTALPHA : {A0}

Figure 36.42. EBI Alpha Value Equation

When the 9-bit alpha blending factor is defined in the EBI_TFTALPHA register is used, the maximum allowed value for ALPHA is 255. If the MSB bit of the EBI_TFTALPHA field is set, the ALPHA use will saturate to 255. An alpha value of 0 corresponds to a fully transparent color, whereas an alpha value of 255 corresponds to a fully opaque color. The RGB Color0 data is taken from either the internal write data (written to EBI_TFTPIXEL0) or from the external write data (written to bank BANKSEL). The Color0 source selection is based on the MASKBLEND bitfield of the EBI_TFTCTRL register. Internal write data is used for MASKBLEND settings equal to IMASK, IALPHA, or IMASKIALPHA. External write data is used for MASKBLEND settings equal to EFBMASK, EFBALPHA, EFBMASKEALPHA, IFBMASK, IFBALPHA or IFBMASKALPHA. In the EFBMASKEALPHA mode, the Color1 operand is read from either the BANKSEL memory bank or from the EBI_TFTPIXEL1 register as defined in the COLOR1SRC bitfield of the EBI_TFTCTRL register. In the IFBMASKALPHA mode, the Color0 operand is read from the BANKSEL memory bank and Color1 operand is the external write data. The alpha blended result will be written to the BANKSEL memory bank for EFBMASKALPHA mode. For IMASKALPHA and IFBMASKALPHA modes, the alpha blended result is stored in the EBI_TFTPIXEL register.

For transactions involving an external memory device, the automatic transaction translation rules as described in 36.3.11 Data Access Width apply. For example, 1 32-bit wide AHB write to a 16-bit wide external memory can be used to automatically perform 2 16-bit alpha blending operations into external memory. Three configurations of data source and destination are supported as described next.

In the scenario,the alpha blending into the external memory frame buffer is performed by writing RGB or ARGB data D to address A in bank BANKSEL with COLOR1SRC set to MEM and MASKBLEND set to EFBMASK, EFBALPHA, EFBMASKEALPHA. Note that in this case the EBI automatically translates the AHB write transaction into a read-modify-write sequence for the external memory.

Memory[A] = AlphaBlend(D, Memory[A])

Figure 36.43. EBI In-place Alpha Blending into External Memory

Alpha blending into external memory with a Color1 from register is performed by writing RGB data D to address A in bank BANKSEL with COLOR1SRC set to PIXEL1 and MASKBLEND set to EFBMASK, EFBALPHA or EFBMASKEALPHA:

Memory[A] = AlphaBlend(D, EBI_TFTPIXEL1)

Figure 36.44. EBI Alpha Blending into External Memory with Background Color1 from Register

For alpha blending to an internal frame buffer, the MASKBLEND field is set to either IFBMASK, IFBALPHA or IFBMASKALPHA. The TFTPIXEL1EMPTY DMA and Interrupt flags will be set to request for a background color. Then the DMA / CPU writes a background RGB Data D to address A in bank BANKSEL to start the blending operation. The foreground color comes from the external memory address A and the result from the blending is written to EBI_TFTPIXEL register. The TFTPIXELFULL flag will be set to inform DMA / CPU to move the result in EBI_TFTPIXEL register to an internal memory location. Refer to Figure 36.45 EBI Alpha Blending into Internal Frame Buffer on page 1334 for the internal frame buffer formula.

EBI_TFTPIXEL = AlphaBlend(Memory[A], D)

Figure 36.45. EBI Alpha Blending into Internal Frame Buffer

Internal alpha blending into register EBI_TFTPIXEL is performed by writing RGB data D to EBI_TFTPIXEL0 with COLOR1SRC set to PIXEL1 and MASKBLEND set to IMASK, IALPHA, or IMASKEALPHA. This alpha blending interface is intended for use by both the Cortex-M4 and the DMA controller. For DMA operation three DMA requests are generated. One DMA request indicating that EBI_TFTPIXEL0 requires new data, one DMA request indicating that EBI_TFTPIXEL1 requires new data, and one DMA request indicating that new blended data is available in EBI_TFTPIXEL. The write into EBI_TFTPIXEL0 triggers the alpha blending operation. If software wants to reprogram EBI_TFTPIXEL1, then this should be done before the EBI_TFTPIXEL0 write, which triggers the alpha blending. The status of the internal alpha blending interface can also be read via the TFTPIXEL0EMPTY, TFTPIXEL1EMPTY, and TFTPIXELFULL bits in the EBI_STATUS register. These 3 status flags are also duplicated as Interrupt Flags. These DMA requests and interrupt flags are also available when in external Alpha blending modes.

EBI_TFTPIXEL = AlphaBlend(EBI_TFTPIXEL0, EBI_TFTPIXEL1)

Figure 36.46. EBI Internal Alpha Blending from Registers into Register

36.3.17.2 Masking

The masking feature can be used to suppress the blending operation and pass the background color as it is. If the EFBMASK or EFBMASKEALPHA modes are used and a write is issued to the external memory. If the write Color0 match the TFTMASK color, the result write operation is suppressed. Whereas in IFBMASK and IFBMASKALPHA modes, a mask match condition will still result in having background color write into the EBI_TFT_PIXEL. This is to prevent mis-alignment of the EBI_TFTPIXEL DMA read pointer from the background DMA write pointer when operating in any of the Internal Frame Buffer modes.

Masking is supported for writes to an external device and for writes to internal register EBI_TFTPIXEL0. Masking in this setting is always based on EBI_TFTMASK data and also depend on the Color0 format defined in the PIXEL0FORMAT bitfield of the EBI_TFTCOLORFORMAT register. In the event when the Color0 ALPHA is 0, the Color1 is also passed unmodified. For transactions involving an external memory device, the automatic transaction translation rules as described in 36.3.11 Data Access Width apply. For example, 1 32-bit wide AHB write to a 16-bit wide external memory can be used to perform masking operations (of color format 16 bits and below) on both 16-bit transactions to the external device. Masking can for example be used when drawing an icon with rounded corners into an external frame buffer. Such an icon can be written to the frame buffer using a 2-dimensional copy action. If the color of a pixel outside the rounded corners is set to match the value defined in the EBI_TFTMASK register, then such a matching data transfer is suppressed. The resulting image in the frame buffer will keep its original background around the corners of the icon.

Internal masking is enabled by setting the MASKBLEND field in the EBI_TFTCTRL register to IMASK or IMASKALPHA. If enabled and EBI_TFTPIXEL0 is written with data matching EBI_TFTMASK, then the background color from EBI_TFTPIXEL1 is copied into EBI_TFTPIXEL. If blend is enabled and EBI_TFTPIXEL0 is written with data not matching EBI_TFTMASK, then the color from EBI_TFTPIXEL0 is alpha blended with EBI_TFTPIXEL1 and result is written into EBI_TFTPIXEL. The three DMA requests, Interrupts Flags and EBI_STATUS bits as described for alpha blending also apply for masking.

36.3.18 Direct Drive Timing

The timing definition for operating a TFT display in Direct Drive mode depends on where the frame buffer source is located. In case internal memory is used as source, then only the TFT timing as defined in the EBI_TFTTIMING register is relevant. In case external memory is used as the source memory, then both the timing parameters of the TFT display and the timing parameters of the memory bank defined in the BANKSEL bitfield of the EBI_TFTCTRL register are relevant.

The minimum dot clock, EBI_DCLK, period is defined in the DCLKPERIOD bitfield of the EBI_TFTTIMING register. This parameter has a minimum duration of 1 cycle, which is set by HW, and writing a value n to this bitfield results in an extended duration of 1+n cycles. At cycle 0 (and then periodically with period DCLKPERIOD + 1) the EBI_DCLK inactive edges are generated. At the cycle defined in the TFTSTART bitfield of the EBI_TFTTIMING the TFT Direct Drive transaction is started. The TFTSTART bitfield can be used to define the duty cycle of the EBI_DCLK. This parameter has a minimum duration of 1 cycle, which is set by HW, and writing a value n to this bitfield results in an extended duration of 1+n cycles. After performing the required actions to produce the required TFT pixel data on the EBI_AD lines, the TFT transaction will pass through its TFTSETUP and TFTHOLD states as indicated in Figure 36.47 EBI TFT Pixel Timing on page 1336. In this figure, the duration of the states in the TFT transaction is defined by the corresponding uppercase name above the state and it is expressed in internal clock cycles. The TFT setup and hold times are set in the TFTHOLD and TFTSETUP bitfields in the EBI_TFTTIMING register. Writing a value m to one of these bitfields results in a duration of the corresponding state of m internal clock cycles. If these parameters are set to 0, it effectively means that the state is skipped. The TFT setup and hold timing is with respect to the active edge of EBI_DCLK as defined in the DCLKPOL bitfield in the EBI_TFTPOLARITY register. The TFT setup and hold timing applies to all TFT signals: EBI_AD, EBI_DATAEN, EBI_VSYNC, EBI_HSYNC and EBI_TFTCSn. The active EBI_DCLK edge is generated in between the TFTSETUP and TFTHOLD states. The TFTSTART bitfield therefore impacts the position of the active EBI_DCLK edge. The later the TFT transaction is started, the later it will transition from its TFTSETUP to TFTHOLD state. If needed, the EBI_DCLK period is automatically stretched beyond the DCLKPERIOD to complete the TFT transaction. EBI_DCLK period stretching occurs when the TFT transaction does not complete in the specified time, which in turn can occur because of the following reasons:

- Specified timing parameters are conflicting. This can for example happen if the TFT setup plus hold time is programmed to be longer than the EBI_DCLK period.
- TFT transaction is delayed by an ongoing EBI transaction. This transaction interference can be controlled by setting the transaction interleaving strategy in the INTERLEAVE bitfield of the EBI_TFTCTRL register.
- TFT transaction data is not delivered in time. For internal Direct Drive this is caused by the Cortex-M4 or DMA not delivering the data in time. For external Direct Drive the timing parameters defining the external device read access might not allow the TFT transaction to complete in time.

In case the specified DCLK_PERIOD is not met, the DDJIT interrupt flag in the EBI_IF register will be set.

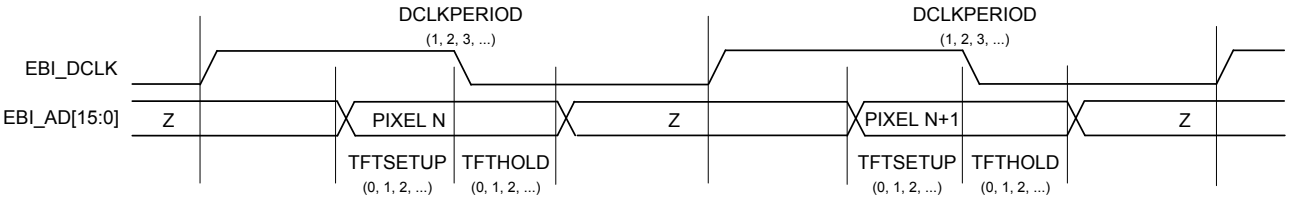


Figure 36.47. EBI TFT Pixel Timing

When driving the TFT from internal memory, the TFT timing is defined in the EBI_TFTTIMING register as shown in Figure 36.48 EBI TFT Direct Drive Internal Timing on page 1336. Before each TFT transaction to the visible part of the display, the EBI will request new pixel data via an interrupt or DMA request. At the time specified in the TFTSTART bitfield of the EBI_TFTTIMING register (and when pixel data has been provided), the TFT transaction will start. For internal Direct Drive the TFT state machine will place the pixel data on the EBI_AD lines during the TFTWDATA state after which the state machine will pass through the programmable TFTSETUP and TFTHOLD states.

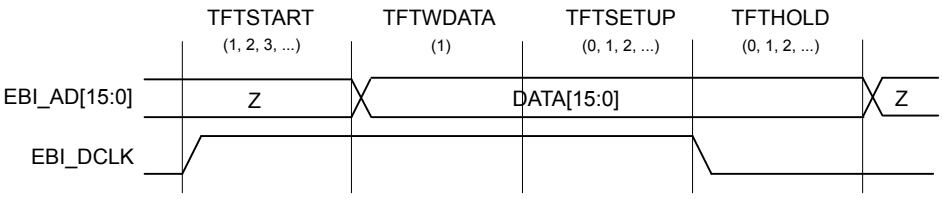


Figure 36.48. EBI TFT Direct Drive Internal Timing

When the TFT is driven directly from an external memory, the timing definitions for the bank defined in the BANKSEL bitfield of the EBI_TFTCTRL register and those for the TFT are both used by Direct Drive to generate transactions satisfying the requirements of both the memory device and the TFT display. The timing definition for the external memory device should be programmed according to its requirements independent of the TFT timing. Figure 36.49 EBI TFT Direct Drive External Timing on page 1337 shows an example of the Direct Drive engine accessing an external memory using the multiplexed 16-bit data, 16-bit address (D16A16ALE) mode. The TFTSETUP and TFTHOLD states are now enclosed within the read transaction states of the chosen mode. The external device read transaction is started at a time as defined by TFTSTART. The read strobe on EBI_REn is automatically extended in duration to satisfy the TFT setup and hold requirements defined in the TFTSETUP and TFTHOLD bitfields.

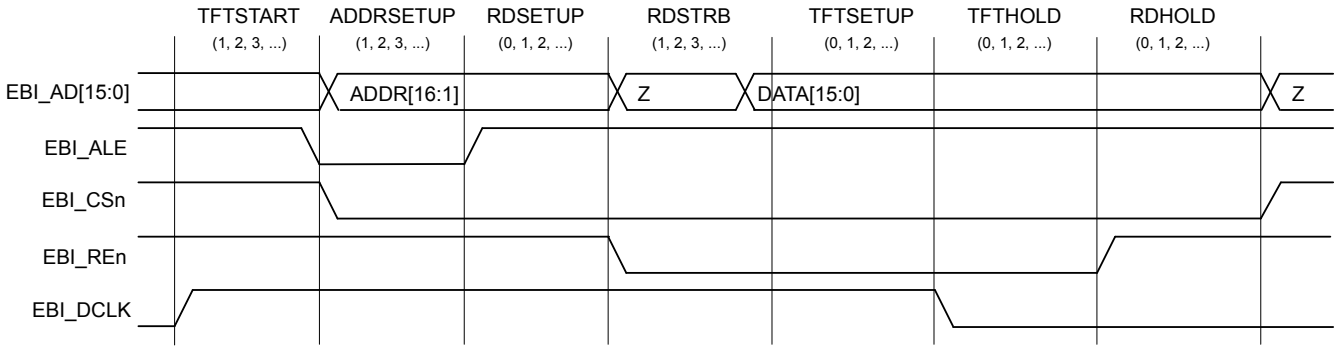


Figure 36.49. EBI TFT Direct Drive External Timing

The timing parameters related to the horizontal timing are shown in Figure 36.50 EBI TFT Horizontal Porch Timing on page 1337. These parameters are defined as pixel or EBI_DCLK counts. The horizontal porch widths are defined in the HBPORCH and HFPORCH bitfields of the EBI_TFTHPORCH register. A porch which has its width parameter programmed to 0 will be skipped. The width and start position of the horizontal synchronization pulse EBI_HSYNC is programmed via the HSYNC and HSYNCSTART bitfields in the EBI_TFTHPORCH register.

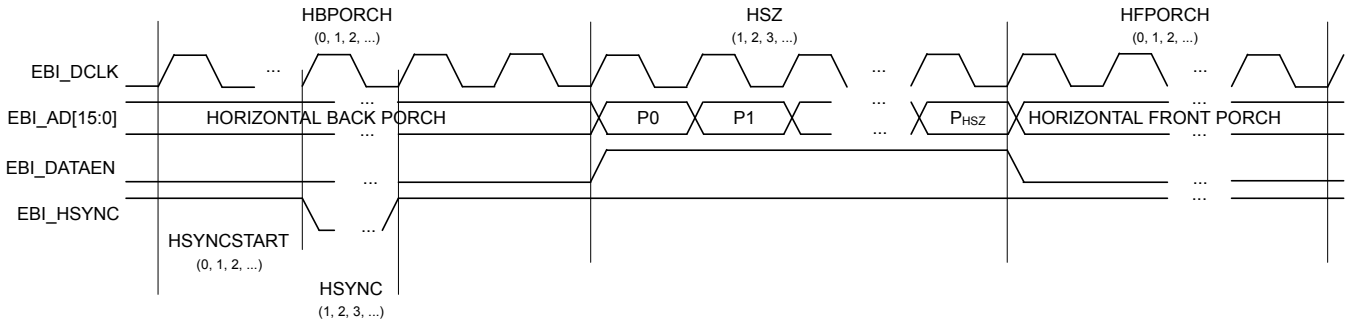


Figure 36.50. EBI TFT Horizontal Porch Timing

The timing parameters related to the vertical timing are shown in Figure 36.51 EBI TFT Vertical Porch Timing on page 1338. These parameters are defined as line or EBI_HSYNC counts. The vertical porch widths are defined in the VBPORCH and VFPORCH bitfields of the EBI_TFTVPORCH register. A porch which has its width parameter programmed to 0 will be skipped. The width of the vertical synchronization pulse EBI_VSYNC is programmed via the VSYNC bitfield in the EBI_TFTVPORCH register.

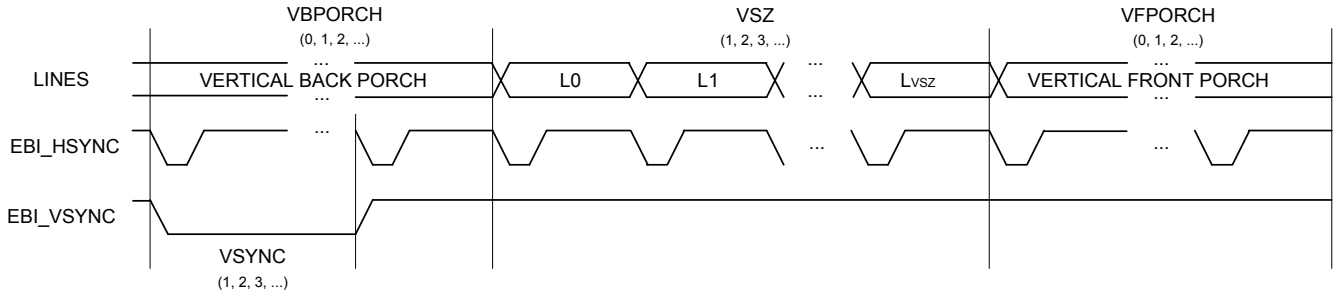


Figure 36.51. EBI TFT Vertical Porch Timing

The active edge of the EBI_DCLK and the other TFT related signals are by default driven off the positive edge of the internal clock. The edges of the EBI_DCLK can also be driven off the negative edge of the internal clock by setting the SHIFTDCLK bitfield in the EBI_TFTCTRL register to 1. The Direct Drive engine then shifts the active DCLK edge 1/2 an internal cycle into the TFTHOLD state. Effectively the length of TFTSETUP state is increased by 1/2 an internal cycle, whereas the length of the TFTHOLD state is decreased by 1/2 an internal cycle. SHIFTDCLK should not be set if TFTHOLD is set to zero cycles. The effect of the SHIFTDCLK bitfield is shown in Figure 36.52 EBI TFT Pixel Timing: EBI_DCLK driven off Positive Edge Internal Clock on page 1338 and Figure 36.53 EBI TFT Pixel Timing: EBI_DCLK driven off Negative Edge Internal Clock on page 1338 for a setup using the falling EBI_DCLK clock as its active edge.

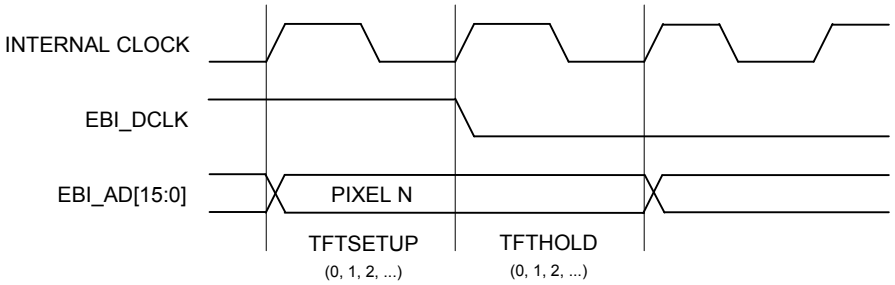


Figure 36.52. EBI TFT Pixel Timing: EBI_DCLK driven off Positive Edge Internal Clock

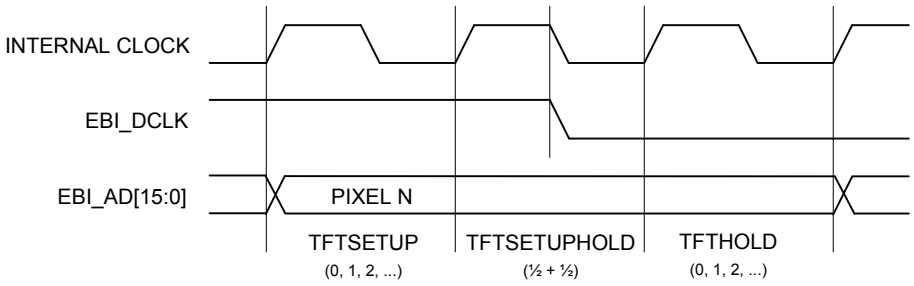


Figure 36.53. EBI TFT Pixel Timing: EBI_DCLK driven off Negative Edge Internal Clock

36.3.19 Control Signal Polarity

It is possible to individually configure the control signals to be active high/low by setting or clearing the appropriate bits in the EBI_POLARITY register. When the ITS bitfield in the EBI_CTRL register is set to 0, the polarities defined in the EBI_POLARITY register applies to all 4 memory banks. When ITS is set to 1 each memory bank uses an individual polarity definition. In this case register EBI_POLARITY only applies to bank 0. Timing for bank n is then defined in the EBI_POLARITYn register.

The TFT control signals can also be individually configured to be active high/low by setting or clearing the appropriate bits in the EBI_TFTPOLARITY register.

36.3.20 Pin Configuration

In order to give the EBI access to the external pins of the EFM32 Giant Gecko 12, the GPIO must be configured accordingly. The lines must be set to Push-Pull, which is described in detail in the GPIO section.

All the EBI pins are enabled in the EBI_ROUTEPEN register. The EBI_AD, EBI_WEn and EBI_REn pins are all enabled by the EBIPEN bit, the EBI_CSn pins are enabled by the corresponding CSxPEN bit, the EBI_ALE pin is enabled by the ALEPEN bit, the EBI_BL pins are enabled by the BLPEN bit, the EBI_NANDWEn and EBI_NANDREn pins are enabled by the NANDPEN bit, the TFT pins EBI_DCLK, EBI_VSYNC and EBI_HSYNC are all enabled by the TFTPEN bit, the EBI_DATAEN pin is enabled by the DATAENPEN bit, the EBI_CSTFT pin is enabled by the CSTFTPEN bit, the EBI_A pins are enabled by the ALB and APEN bitfields, and the EBI_ARDY pin is enabled by the ARDYPEN bit.

For EBI_WEn, EBI_REn, EBI_BL, EBI_ARDY and EBI_ALE pins, alternative pin locations can be chosen by setting the EBILOC bitfield in the EBI_ROUTELOC0 register. The EBI_CS pins alternative location is set by the CSLOC bitfield. The NANDREn and NANDWEn pins alternative pin locations can be set in the NANDLOC bitfield. The EBI_DCLK, EBI_VSYNC, EBI_HSYNC, EBI_DATAEN and CSTFT pins alternative location settings are determined by the TFTLOC bitfield.

The address pins EBI_A alternative locations are determined by the ALOC bitfield value in the EBI_ROUTELOC1 register. While the ADLOC bitfield determined the alternative locations of the EBI_AD pins locations. Details of these alternative locations are specified in the data sheet.

36.3.21 Interrupts

The TFT controller has 6 separate interrupt flags (VSYNC, HSYNC, VBPORCH, VFPORCH, DDEMPTY, DDJIT) in EBI_IF.

The VSYNC, HSYNC, VBPORCH, and VFPORCH interrupt flags indicate various synchronization points during the display of a frame. Figure 36.54 EBI TFT Interrupts on page 1340 shows the timing of the VSYNC, HSYNC, VBPORCH, and VFPORCH interrupt flags. The VSYNC and HSYNC flags are set at the beginning of a frame and at the beginning of a line respectively. The VBPORCH and VFPORCH flags are set at the end of the vertical back porch and at the beginning of the vertical front porch respectively (provided that the related porch is defined with a non-zero width).

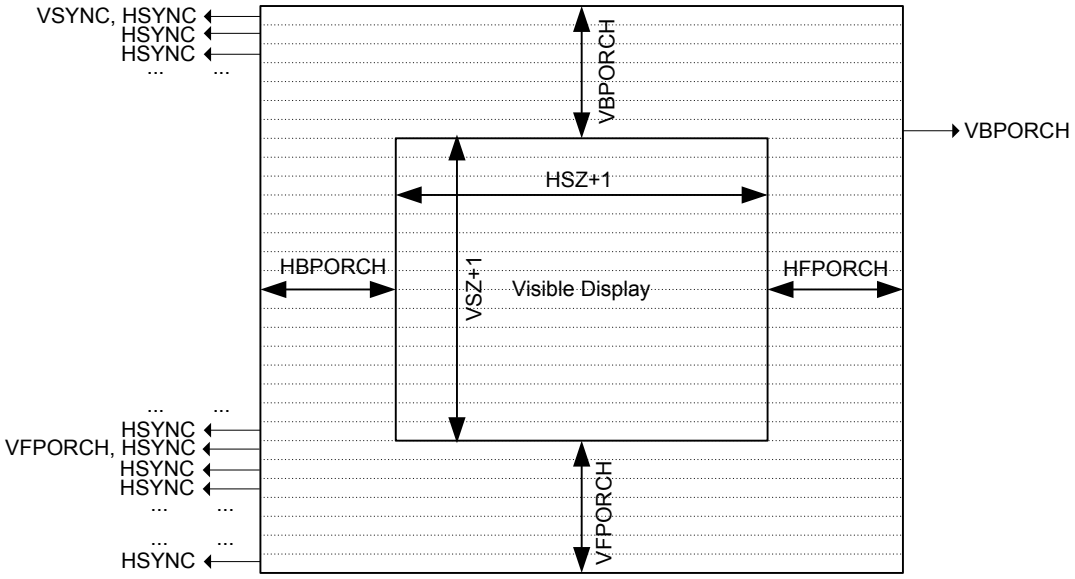


Figure 36.54. EBI TFT Interrupts

The DDEMPTY interrupt flag indicates that the EBI_TFTDD register is empty during Direct Drive from internal memory. The DDJIT interrupt flag indicates that the DCLKPERIOD is not met during Direct Drive operation.

Setting one of the interrupt flags will result in an EBI interrupt if the corresponding interrupt enable bit is set in the EBI_IEN register. All generated interrupts from the EBI will activate the same interrupt vector when enabled.

36.3.22 DMA Request

In internal Direct Drive mode, when the DD bitfield in EBI_TFTCTRL register is INTERNAL, the TFT controller sends out a DMA request when the pixel buffer EBI_TFTDD is empty and needs to be filled. This request is initially set and it is cleared when EBI_TFTDD is written. It is set again once the pixel data has been transferred to the display. One DMA request is generated for each visible pixel.

The masking and alpha blending hardware uses three DMA requests related to the status of thee internal masking and alpha blending registers EBI_TFTPIXEL0, EBI_TFTPIXEL1, and EBI_TFTPIXEL. The DMA request for EBI_TFTPIXEL0 indicates that new data can be written to be used for internal masking or alpha blending. This request is initially set and it is cleared when EBI_TFTPIXEL0 is written. The request is set again when EBI_TFTPIXEL is read. The DMA request for EBI_TFTPIXEL1 is initially set and it is cleared when EBI_TFTPIXEL1 is written. Only when both EBI_TFTPIXEL0 and EBI_TFTPIXEL1 have been written, will a EBI_TFTPIXEL read set the DMA request for EBI_TFTPIXEL1 again. The DMA request for EBI_TFTPIXEL indicates whether new masked and/or blended data is available for reading in EBI_TFTPIXEL or not. It is set after completion of internal masking and alpha blending in reaction to a write to EBI_TFTPIXEL0. It is cleared when EBI_TFTPIXEL is read.

36.4 Register Map

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

Offset	Name	Type	Description
0x000	EBI_CTRL	RW	Control Register
0x004	EBI_ADDRTIMING	RW	Address Timing Register
0x008	EBI_RDTIMING	RW	Read Timing Register
0x00C	EBI_WRTIMING	RW	Write Timing Register
0x010	EBI_POLARITY	RW	Polarity Register
0x018	EBI_ADDRTIMING1	RW	Address Timing Register 1
0x01C	EBI_RDTIMING1	RW	Read Timing Register 1
0x020	EBI_WRTIMING1	RW	Write Timing Register 1
0x024	EBI_POLARITY1	RW	Polarity Register 1
0x028	EBI_ADDRTIMING2	RW	Address Timing Register 2
0x02C	EBI_RDTIMING2	RW	Read Timing Register 2
0x030	EBI_WRTIMING2	RW	Write Timing Register 2
0x034	EBI_POLARITY2	RW	Polarity Register 2
0x038	EBI_ADDRTIMING3	RW	Address Timing Register 3
0x03C	EBI_RDTIMING3	RW	Read Timing Register 3
0x040	EBI_WRTIMING3	RW	Write Timing Register 3
0x044	EBI_POLARITY3	RW	Polarity Register 3
0x048	EBI_PAGECTRL	RW	Page Control Register
0x04C	EBI_NANDCTRL	RW	NAND Control Register
0x050	EBI_CMD	W1	Command Register
0x054	EBI_STATUS	R	Status Register
0x058	EBI_ECCPARITY	R	ECC Parity Register
0x05C	EBI_TFTCTRL	RW	TFT Control Register
0x060	EBI_TFTSTATUS	R	TFT Status Register
0x064	EBI_TFTCOLORFORMAT	RW	Color Format Register
0x068	EBI_TFTFRAMEBASE	RW	TFT Frame Base Register
0x070	EBI_TFTSTRIDE	RW	TFT Stride Register
0x074	EBI_TFTSIZE	RW	TFT Size Register
0x078	EBI_TFTHPORCH	RW	TFT Horizontal Porch Register
0x07C	EBI_TFTVPORCH	RW	TFT Vertical Porch Register
0x080	EBI_TFTTIMING	RW	TFT Timing Register
0x084	EBI_TFTPOLARITY	RW	TFT Polarity Register
0x088	EBI_TFTDD	RW	TFT Direct Drive Data Register
0x08C	EBI_TFTALPHA	RW	TFT Alpha Blending Register
0x090	EBI_TFTPIXEL0	RW	TFT Pixel 0 Register

Offset	Name	Type	Description
0x094	EBI_TFTPIXEL1	RW	TFT Pixel 1 Register
0x098	EBI_TFTPIXEL	R	TFT Alpha Blending Result Pixel Register
0x09C	EBI_TFTMASK	RW	TFT Masking Register
0x0A0	EBI_IF	R	Interrupt Flag Register
0x0A4	EBI_IFS	W1	Interrupt Flag Set Register
0x0A8	EBI_IFC	(R)W1	Interrupt Flag Clear Register
0x0AC	EBI_IEN	RW	Interrupt Enable Register
0x0B0	EBI_ROUTEPEN	RW	I/O Routing Register
0x0B4	EBI_ROUTELOC0	RW	I/O Routing Location Register
0x0B8	EBI_ROUTELOC1	RW	I/O Routing Location Register

36.5 Register Description

36.5.1 EBI_CTRL - 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	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	
Name	ALTMAP	ITS			BL3	BL2	BL1	BL	ARDYTO3DIS	ARDY3EN	ARDYTO2DIS	ARDY2EN	ARDYTO1DIS	ARDY1EN	ARDYTODIS	ARDYEN	NOIDLE3	NOIDLE2	NOIDLE1	NOIDLE	BANK3EN	BANK2EN	BANK1EN	BANK0EN	MODE3		MODE2		MODE1		MODE	

Bit	Name	Reset	Access	Description
31	ALTMAP	0	RW	**Alternative Address Map Enable**
	This field enables or disables the alternative (256 MB per bank) address map.			
30	ITS	0	RW	**Individual Timing Set, Line Polarity and Mode Definition Enable**
	This field enables or disables individual timing sets, line polarities and modes per bank.			
29:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	BL3	0	RW	**Byte Lane Enable for Bank 3**
	Enables or disables the Byte Lane functionality for bank 3. Ignored when ITS = 0.			
26	BL2	0	RW	**Byte Lane Enable for Bank 2**
	Enables or disables the Byte Lane functionality for bank 2. Ignored when ITS = 0.			
25	BL1	0	RW	**Byte Lane Enable for Bank 1**
	Enables or disables the Byte Lane functionality for bank 1. Ignored when ITS = 0.			
24	BL	0	RW	**Byte Lane Enable for Bank 0**
	Enables or disables the Byte Lane functionality for bank 0. Applies to all banks when ITS = 0. Applies to only bank 0 when ITS = 1.			
23	ARDYTO3DIS	0	RW	**ARDY Timeout Disable for Bank 3**
	Enables or disables the ARDY timeout functionality for bank 3. The timeout value is 32 internal clock cycles. Ignored when ITS = 0.			
22	ARDY3EN	0	RW	**ARDY Enable for Bank 3**
	Enables or disables the ARDY functionality for bank 3. Ignored when ITS = 0.			
21	ARDYTO2DIS	0	RW	**ARDY Timeout Disable for Bank 2**
	Enables or disables the ARDY timeout functionality for bank 2. The timeout value is 32 internal clock cycles. Ignored when ITS = 0.			
20	ARDY2EN	0	RW	**ARDY Enable for Bank 2**
	Enables or disables the ARDY functionality for bank 2. Ignored when ITS = 0.			
19	ARDYTO1DIS	0	RW	**ARDY Timeout Disable for Bank 1**
	Enables or disables the ARDY timeout functionality for bank 1. The timeout value is 32 internal clock cycles. Ignored when ITS = 0.			

Bit	Name	Reset	Access	Description
18	ARDY1EN	0	RW	**ARDY Enable for Bank 1**
	Enables or disables the ARDY functionality for bank 1. Ignored when ITS = 0.			
17	ARDYTODIS	0	RW	**ARDY Timeout Disable**
	Enables or disables the ARDY timeout functionality. The timeout value is 32 internal clock cycles. Applies to all banks when ITS = 0. Applies to only bank 0 when ITS = 1.			
16	ARDYEN	0	RW	**ARDY Enable**
	Enables or disables the ARDY functionality. Applies to all banks when ITS = 0. Applies to only bank 0 when ITS = 1.			
15	NOIDLE3	0	RW	**No Idle Cycle Insertion on Bank 3**
	Enables or disables idle state insertion between transfers for bank 3. Ignored when ITS = 0.			
14	NOIDLE2	0	RW	**No Idle Cycle Insertion on Bank 2**
	Enables or disables idle state insertion between transfers for bank 2. Ignored when ITS = 0.			
13	NOIDLE1	0	RW	**No Idle Cycle Insertion on Bank 1**
	Enables or disables idle state insertion between transfers for bank 1. Ignored when ITS = 0.			
12	NOIDLE	0	RW	**No Idle Cycle Insertion on Bank 0**
	Enables or disables idle state insertion between transfers for bank 0. Applies to all banks when ITS = 0. Applies to only bank 0 when ITS = 1.			
11	BANK3EN	0	RW	**Bank 3 Enable**
	This field enables or disables bank 3.			
10	BANK2EN	0	RW	**Bank 2 Enable**
	This field enables or disables bank 2.			
9	BANK1EN	0	RW	**Bank 1 Enable**
	This field enables or disables bank 1.			
8	BANK0EN	0	RW	**Bank 0 Enable**
	This field enables or disables bank 0.			
7:6	MODE3	0x0	RW	**Mode 3**
	This field sets the access mode the EBI will use for interfacing devices on bank 3. Ignored when ITS = 0.			

Value	Mode	Description
0	D8A8	EBI_AD drives 8 bit data, 8 bit address, ALE not used. Extended address bits can be enabled.
1	D16A16ALE	EBI_AD drives 16 bit data, 16 bit address, ALE is used for address latching. Extended address bits can be enabled.
2	D8A24ALE	EBI_AD drives 8 bit data, 24 bit address, ALE is used for address latching. Extended address bits can be enabled.
3	D16	EBI_AD drives 16 bit data, ALE not used. Extended address bits can be enabled.

5:4	MODE2	0x0	RW	**Mode 2**
	This field sets the access mode the EBI will use for interfacing devices on bank 2. Ignored when ITS = 0.			

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	D8A8		EBI_AD drives 8 bit data, 8 bit address, ALE not used. Extended address bits can be enabled.
	1	D16A16ALE		EBI_AD drives 16 bit data, 16 bit address, ALE is used for address latching. Extended address bits can be enabled.
	2	D8A24ALE		EBI_AD drives 8 bit data, 24 bit address, ALE is used for address latching. Extended address bits can be enabled.
	3	D16		EBI_AD drives 16 bit data, ALE not used. Extended address bits can be enabled.
3:2	MODE1	0x0	RW	**Mode 1**
	This field sets the access mode the EBI will use for interfacing devices on bank 1. Ignored when ITS = 0.			
	Value	Mode		Description
	0	D8A8		EBI_AD drives 8 bit data, 8 bit address, ALE not used. Extended address bits can be enabled.
	1	D16A16ALE		EBI_AD drives 16 bit data, 16 bit address, ALE is used for address latching. Extended address bits can be enabled.
	2	D8A24ALE		EBI_AD drives 8 bit data, 24 bit address, ALE is used for address latching. Extended address bits can be enabled.
	3	D16		EBI_AD drives 16 bit data, ALE not used. Extended address bits can be enabled.
1:0	MODE	0x0	RW	**Mode**
	This field sets the access mode the EBI will use for interfacing devices. Applies to all banks when ITS = 0. Applies to only bank 0 when ITS = 1.			
	Value	Mode		Description
	0	D8A8		EBI_AD drives 8 bit data, 8 bit address, ALE not used. Extended address bits can be enabled.
	1	D16A16ALE		EBI_AD drives 16 bit data, 16 bit address, ALE is used for address latching. Extended address bits can be enabled.
	2	D8A24ALE		EBI_AD drives 8 bit data, 24 bit address, ALE is used for address latching. Extended address bits can be enabled.
	3	D16		EBI_AD drives 16 bit data, ALE not used. Extended address bits can be enabled.

36.5.2 EBI_ADDRTIMING - Address Timing 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				0																		0x7								0x7		
Access				RW																		RW								RW		
Name				HALFALE																		ADDRHOLD								ADDRSETUP		

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	HALFALE	0	RW	**Half Cycle ALE Strobe Duration Enable**
	Enables or disables half cycle duration of the ALE strobe in the last ADDRSETUP cycle.			
27:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	ADDRHOLD	0x7	RW	**Address Hold Time**
	Sets the number of cycles the address is held after ALE is asserted.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	ADDRSETUP	0x7	RW	**Address Setup Time**
	Sets the number of cycles the address is driven onto the ADDRDAT bus before ALE is asserted. If set to 0, 1 cycle is inserted by HW.			

36.5.3 EBI_RDTIMING - Read Timing 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		0	0	0										0x7				0x7F												0x7		
Access		RW	RW	RW										RW				RW												RW		
Name		PAGEMODE	PREFETCH	HALFRE										RDHOLD				RDSTRB												RDSETUP		

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	PAGEMODE	0	RW	**Page Mode Access Enable**
	Enables or disables page mode reads.			
29	PREFETCH	0	RW	**Prefetch Enable**
	Enables or disables prefetching of data from sequential address.			
28	HALFRE	0	RW	**Half Cycle REn Strobe Duration Enable**
	Enables or disables half cycle duration of the REn strobe in the last RDSTRB cycle.			
27:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	RDHOLD	0x7	RW	**Read Hold Time**
	Sets the number of cycles CSn is held active after the REn is deasserted. This interval is used for bus turnaround.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	RDSTRB	0x7F	RW	**Read Strobe Time**
	Sets the number of cycles the REn is held active. After the specified number of cycles, data is read. If set to 0, 1 cycle is inserted by HW.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	RDSETUP	0x7	RW	**Read Setup Time**
	Sets the number of cycles the address setup before REn is asserted.			

36.5.4 EBI_WRTIMING - Write Timing 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			0	0										0x7				0x7F												0x7		
Access			RW	RW										RW				RW												RW		
Name			WBUFDIS	HALFWE										WRHOLD				WRSTRB												WRSETUP		

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	WBUFDIS	0	RW	**Write Buffer Disable**
	Enables or disables the write buffer.			
28	HALFWE	0	RW	**Half Cycle WEn Strobe Duration Enable**
	Enables or disables half cycle duration of the WEn strobe in the last WRSTRB cycle.			
27:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	WRHOLD	0x7	RW	**Write Hold Time**
	Sets the number of cycles CSn is held active after the WEn is deasserted.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	WRSTRB	0x7F	RW	**Write Strobe Time**
	Sets the number of cycles the WEn is held active. If set to 0, 1 cycle is inserted by HW.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	WRSETUP	0x7	RW	**Write Setup Time**
	Sets the number of cycles the address setup before WEn is asserted.			

36.5.5 EBI_POLARITY - Polarity 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																											0	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											BLPOL	ARDYPOL	ALEPOL	WEPOL	REPOL	CSPOL

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	BLPOL	0	RW	**BL Polarity**

Sets the polarity of the EBI_BLn lines.

Value	Mode	Description
0	ACTIVELOW	BLn[1:0] are active low.
1	ACTIVEHIGH	BLn[1:0] are active high.

Bit	Name	Reset	Access	Description
4	ARDYPOL	0	RW	**ARDY Polarity**

Sets the polarity of the EBI_ARDY line.

Value	Mode	Description
0	ACTIVELOW	ARDY is active low.
1	ACTIVEHIGH	ARDY is active high.

Bit	Name	Reset	Access	Description
3	ALEPOL	0	RW	**Address Latch Polarity**

Sets the polarity of the EBI_ALE line.

Value	Mode	Description
0	ACTIVELOW	ALE is active low.
1	ACTIVEHIGH	ALE is active high.

Bit	Name	Reset	Access	Description
2	WEPOL	0	RW	**Write Enable Polarity**

Sets the polarity of the EBI_WEn and EBI_NANDWEn lines.

Value	Mode	Description
0	ACTIVELOW	WEn and NANDWEn are active low.
1	ACTIVEHIGH	WEn and NANDWEn are active high.

Bit	Name	Reset	Access	Description
1	REPOL	0	RW	**Read Enable Polarity**

Sets the polarity of the EBI_REn and EBI_NANDREn lines.

Value	Mode	Description
0	ACTIVELOW	REn and NANDREn are active low.

Bit	Name	Reset	Access	Description
	1	ACTIVEHIGH		REn and NANDREn are active high.
0	CSPOL	0	RW	**Chip Select Polarity**
	Sets the polarity of the EBI_CSn line.			

Value	Mode	Description
0	ACTIVELOW	CSn is active low.
1	ACTIVEHIGH	CSn is active high.

36.5.6 EBI_ADDRTIMING1 - Address Timing Register 1

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				0																		0x7								0x7	
Access				RW																		RW								RW	
Name				HALFALE																		ADDRHOLD								ADDRSETUP	

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	HALFALE	0	RW	**Half Cycle ALE Strobe Duration Enable**
	Enables or disables half cycle duration of the ALE strobe in the last address setup cycle.			
27:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	ADDRHOLD	0x7	RW	**Address Hold Time**
	Sets the number of cycles the address is held after ALE is asserted.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	ADDRSETUP	0x7	RW	**Address Setup Time**
	Sets the number of cycles the address is driven onto the ADDRDAT bus before ALE is asserted. If set to 0, 1 cycle is inserted by HW.			

36.5.7 EBI_RDTIMING1 - Read Timing Register 1

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		0	0	0										0x7				0x7F												0x7		
Access		RW	RW	RW										RW				RW												RW		
Name		PAGEMODE	PREFETCH	HALFRE										RDHOLD				RDSTRB												RDSETUP		

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	PAGEMODE	0	RW	**Page Mode Access Enable**
	Enables or disables page mode reads.			
29	PREFETCH	0	RW	**Prefetch Enable**
	Enables or disables prefetching of data from sequential address.			
28	HALFRE	0	RW	**Half Cycle REn Strobe Duration Enable**
	Enables or disables half cycle duration of the REn strobe in the last RDSTRB cycle.			
27:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	RDHOLD	0x7	RW	**Read Hold Time**
	Sets the number of cycles CSn is held active after the REn is deasserted. This interval is used for bus turnaround.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	RDSTRB	0x7F	RW	**Read Strobe Time**
	Sets the number of cycles the REn is held active. After the specified number of cycles, data is read. If set to 0, 1 cycle is inserted by HW.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	RDSETUP	0x7	RW	**Read Setup Time**
	Sets the number of cycles the address setup before REn is asserted.			

36.5.8 EBI_WRTIMING1 - Write Timing 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			0	0										0x7				0x7F												0x7		
Access			RW	RW										RW				RW												RW		
Name			WBUFDIS	HALFWE										WRHOLD				WRSTRB												WRSETUP		

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	WBUFDIS	0	RW	**Write Buffer Disable**
	Enables or disables the write buffer.			
28	HALFWE	0	RW	**Half Cycle WEn Strobe Duration Enable**
	Enables or disables half cycle duration of the WEn strobe in the last WRSTRB cycle.			
27:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	WRHOLD	0x7	RW	**Write Hold Time**
	Sets the number of cycles CSn is held active after the WEn is deasserted.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	WRSTRB	0x7F	RW	**Write Strobe Time**
	Sets the number of cycles the WEn is held active. If set to 0, 1 cycle is inserted by HW.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	WRSETUP	0x7	RW	**Write Setup Time**
	Sets the number of cycles the address setup before WEn is asserted.			

36.5.9 EBI_POLARITY1 - Polarity Register 1

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																											0	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											BLPOL	ARDYPOL	ALEPOL	WEPOL	REPOL	CSPOL

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	BLPOL	0	RW	**BL Polarity**

Sets the polarity of the EBI_BLn lines.

Value	Mode	Description
0	ACTIVELOW	BLn[1:0] are active low.
1	ACTIVEHIGH	BLn[1:0] are active high.

Bit	Name	Reset	Access	Description
4	ARDYPOL	0	RW	**ARDY Polarity**

Sets the polarity of the EBI_ARDY line.

Value	Mode	Description
0	ACTIVELOW	ARDY is active low.
1	ACTIVEHIGH	ARDY is active high.

Bit	Name	Reset	Access	Description
3	ALEPOL	0	RW	**Address Latch Polarity**

Sets the polarity of the EBI_ALE line.

Value	Mode	Description
0	ACTIVELOW	ALE is active low.
1	ACTIVEHIGH	ALE is active high.

Bit	Name	Reset	Access	Description
2	WEPOL	0	RW	**Write Enable Polarity**

Sets the polarity of the EBI_WEn and EBI_NANDWEn lines.

Value	Mode	Description
0	ACTIVELOW	WEn and NANDWEn are active low.
1	ACTIVEHIGH	WEn and NANDWEn are active high.

Bit	Name	Reset	Access	Description
1	REPOL	0	RW	**Read Enable Polarity**

Sets the polarity of the EBI_REn and EBI_NANDREn lines.

Value	Mode	Description
0	ACTIVELOW	REn and NANDREn are active low.

Bit	Name	Reset	Access	Description
	1	ACTIVEHIGH		REn and NANDREn are active high.
0	CSPOL	0	RW	**Chip Select Polarity**
	Sets the polarity of the EBI_CSn line.			

Value	Mode	Description
0	ACTIVELOW	CSn is active low.
1	ACTIVEHIGH	CSn is active high.

36.5.10 EBI_ADDRTIMING2 - Address Timing Register 2

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				0																		0x7								0x7		
Access				RW																		RW								RW		
Name				HALFALE																		ADDRHOLD								ADDRSETUP		

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	HALFALE	0	RW	**Half Cycle ALE Strobe Duration Enable**
	Enables or disables half cycle duration of the ALE strobe in the last address setup cycle.			
27:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	ADDRHOLD	0x7	RW	**Address Hold Time**
	Sets the number of cycles the address is held after ALE is asserted.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	ADDRSETUP	0x7	RW	**Address Setup Time**
	Sets the number of cycles the address is driven onto the ADDRDAT bus before ALE is asserted. If set to 0, 1 cycle is inserted by HW.			

36.5.11 EBI_RDTIMING2 - Read Timing Register 2

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		0	0	0										0x7				0x7F												0x7		
Access		RW	RW	RW										RW				RW												RW		
Name		PAGEMODE	PREFETCH	HALFRE										RDHOLD				RDSTRB												RDSETUP		

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	PAGEMODE	0	RW	**Page Mode Access Enable**
	Enables or disables page mode reads.			
29	PREFETCH	0	RW	**Prefetch Enable**
	Enables or disables prefetching of data from sequential address.			
28	HALFRE	0	RW	**Half Cycle REn Strobe Duration Enable**
	Enables or disables half cycle duration of the REn strobe in the last RDSTRB cycle.			
27:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	RDHOLD	0x7	RW	**Read Hold Time**
	Sets the number of cycles CSn is held active after the REn is deasserted. This interval is used for bus turnaround.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	RDSTRB	0x7F	RW	**Read Strobe Time**
	Sets the number of cycles the REn is held active. After the specified number of cycles, data is read. If set to 0, 1 cycle is inserted by HW.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	RDSETUP	0x7	RW	**Read Setup Time**
	Sets the number of cycles the address setup before REn is asserted.			

36.5.12 EBI_WRTIMING2 - Write Timing Register 2

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			0	0										0x7				0x7F												0x7		
Access			RW	RW										RW				RW												RW		
Name			WBUFDIS	HALFWE										WRHOLD				WRSTRB												WRSETUP		

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	WBUFDIS	0	RW	**Write Buffer Disable**
	Enables or disables the write buffer.			
28	HALFWE	0	RW	**Half Cycle WEn Strobe Duration Enable**
	Enables or disables half cycle duration of the WEn strobe in the last WRSTRB cycle.			
27:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	WRHOLD	0x7	RW	**Write Hold Time**
	Sets the number of cycles CSn is held active after the WEn is deasserted.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	WRSTRB	0x7F	RW	**Write Strobe Time**
	Sets the number of cycles the WEn is held active. If set to 0, 1 cycle is inserted by HW.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	WRSETUP	0x7	RW	**Write Setup Time**
	Sets the number of cycles the address setup before WEn is asserted.			

36.5.13 EBI_POLARITY2 - Polarity Register 2

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																											0	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											BLPOL	ARDYPOL	ALEPOL	WEPOL	REPOL	CSPOL

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	BLPOL	0	RW	**BL Polarity**

Sets the polarity of the EBI_BLn lines.

Value	Mode	Description
0	ACTIVELOW	BLn[1:0] are active low.
1	ACTIVEHIGH	BLn[1:0] are active high.

Bit	Name	Reset	Access	Description
4	ARDYPOL	0	RW	**ARDY Polarity**

Sets the polarity of the EBI_ARDY line.

Value	Mode	Description
0	ACTIVELOW	ARDY is active low.
1	ACTIVEHIGH	ARDY is active high.

Bit	Name	Reset	Access	Description
3	ALEPOL	0	RW	**Address Latch Polarity**

Sets the polarity of the EBI_ALE line.

Value	Mode	Description
0	ACTIVELOW	ALE is active low.
1	ACTIVEHIGH	ALE is active high.

Bit	Name	Reset	Access	Description
2	WEPOL	0	RW	**Write Enable Polarity**

Sets the polarity of the EBI_WEn and EBI_NANDWEn lines.

Value	Mode	Description
0	ACTIVELOW	WEn and NANDWEn are active low.
1	ACTIVEHIGH	WEn and NANDWEn are active high.

Bit	Name	Reset	Access	Description
1	REPOL	0	RW	**Read Enable Polarity**

Sets the polarity of the EBI_REn and EBI_NANDREn lines.

Value	Mode	Description
0	ACTIVELOW	REn and NANDREn are active low.

Bit	Name	Reset	Access	Description
	1	ACTIVEHIGH		REn and NANDREn are active high.
0	CSPOL	0	RW	**Chip Select Polarity**
	Sets the polarity of the EBI_CSn line.			

Value	Mode	Description
0	ACTIVELOW	CSn is active low.
1	ACTIVEHIGH	CSn is active high.

36.5.14 EBI_ADDRTIMING3 - Address Timing Register 3

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				0																		0x7								0x7		
Access				RW																		RW								RW		
Name				HALFALE																		ADDRHOLD								ADDRSETUP		

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	HALFALE	0	RW	**Half Cycle ALE Strobe Duration Enable**
	Enables or disables half cycle duration of the ALE strobe in the last address setup cycle.			
27:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	ADDRHOLD	0x7	RW	**Address Hold Time**
	Sets the number of cycles the address is held after ALE is asserted.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	ADDRSETUP	0x7	RW	**Address Setup Time**
	Sets the number of cycles the address is driven onto the ADDRDAT bus before ALE is asserted. If set to 0, 1 cycle is inserted by HW.			

36.5.15 EBI_RDTIMING3 - Read Timing Register 3

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		0	0	0										0x7				0x7F												0x7		
Access		RW	RW	RW										RW				RW												RW		
Name		PAGEMODE	PREFETCH	HALFRE										RDHOLD				RDSTRB												RDSETUP		

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	PAGEMODE	0	RW	**Page Mode Access Enable**
	Enables or disables page mode reads.			
29	PREFETCH	0	RW	**Prefetch Enable**
	Enables or disables prefetching of data from sequential address.			
28	HALFRE	0	RW	**Half Cycle REn Strobe Duration Enable**
	Enables or disables half cycle duration of the REn strobe in the last RDSTRB cycle.			
27:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	RDHOLD	0x7	RW	**Read Hold Time**
	Sets the number of cycles CSn is held active after the REn is deasserted. This interval is used for bus turnaround.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	RDSTRB	0x7F	RW	**Read Strobe Time**
	Sets the number of cycles the REn is held active. After the specified number of cycles, data is read. If set to 0, 1 cycle is inserted by HW.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	RDSETUP	0x7	RW	**Read Setup Time**
	Sets the number of cycles the address setup before REn is asserted.			

36.5.16 EBI_WRTIMING3 - Write Timing Register 3

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			0	0										0x7				0x7F												0x7		
Access			RW	RW										RW				RW												RW		
Name			WBUFDIS	HALFWE										WRHOLD				WRSTRB												WRSETUP		

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	WBUFDIS	0	RW	**Write Buffer Disable**
	Enables or disables the write buffer.			
28	HALFWE	0	RW	**Half Cycle WEn Strobe Duration Enable**
	Enables or disables half cycle duration of the WEn strobe in the last WRSTRB cycle.			
27:19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	WRHOLD	0x7	RW	**Write Hold Time**
	Sets the number of cycles CSn is held active after the WEn is deasserted.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:8	WRSTRB	0x7F	RW	**Write Strobe Time**
	Sets the number of cycles the WEn is held active. If set to 0, 1 cycle is inserted by HW.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	WRSETUP	0x7	RW	**Write Setup Time**
	Sets the number of cycles the address setup before WEn is asserted.			

36.5.17 EBI_POLARITY3 - Polarity Register 3

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																											0	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											BLPOL	ARDYPOL	ALEPOL	WEPOL	REPOL	CSPOL

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	BLPOL	0	RW	**BL Polarity**

Sets the polarity of the EBI_BLn lines.

Value	Mode	Description
0	ACTIVELOW	BLn[1:0] are active low.
1	ACTIVEHIGH	BLn[1:0] are active high.

Bit	Name	Reset	Access	Description
4	ARDYPOL	0	RW	**ARDY Polarity**

Sets the polarity of the EBI_ARDY line.

Value	Mode	Description
0	ACTIVELOW	ARDY is active low.
1	ACTIVEHIGH	ARDY is active high.

Bit	Name	Reset	Access	Description
3	ALEPOL	0	RW	**Address Latch Polarity**

Sets the polarity of the EBI_ALE line.

Value	Mode	Description
0	ACTIVELOW	ALE is active low.
1	ACTIVEHIGH	ALE is active high.

Bit	Name	Reset	Access	Description
2	WEPOL	0	RW	**Write Enable Polarity**

Sets the polarity of the EBI_WEn and EBI_NANDWEn lines.

Value	Mode	Description
0	ACTIVELOW	WEn and NANDWEn are active low.
1	ACTIVEHIGH	WEn and NANDWEn are active high.

Bit	Name	Reset	Access	Description
1	REPOL	0	RW	**Read Enable Polarity**

Sets the polarity of the EBI_REn and EBI_NANDREn lines.

Value	Mode	Description
0	ACTIVELOW	REn and NANDREn are active low.

Bit	Name	Reset	Access	Description
	1	ACTIVEHIGH		REn and NANDREn are active high.
0	CSPOL	0	RW	**Chip Select Polarity**
	Sets the polarity of the EBI_CSn line.			
	Value	Mode		Description
	0	ACTIVELOW		CSn is active low.
	1	ACTIVEHIGH		CSn is active high.

36.5.18 EBI_PAGECTRL - Page 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							0x00															0xF						0			0x0	
Access							RW															RW						RW			RW	
Name							KEEPOPEN															RDPA						INCHIT			PAGELEN	

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*		
26:20	KEEPOPEN	0x00	RW	**Maximum Page Open Time**
	Sets the maximum number of consecutive cycles a page can be considered open. Needs to be larger than 0 in order to be able to benefit from RDPA timing.			
19:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	RDPA	0xF	RW	**Page Read Access Time**
	Sets the number of cycles needed for intrapage page access time. If set to 0, 1 cycle is inserted by HW.			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	INCHIT	0	RW	**Intrapage Hit Only on Incremental Addresses**
	Sets whether page hits occur on any member in a page or only on incremental addresses.			
3:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	PAGELEN	0x0	RW	**Page Length**
	Sets the page length.			

Value	Mode	Description
0	MEMBER4	4 members in a page.
1	MEMBER8	8 members in a page.
2	MEMBER16	16 members in a page.
3	MEMBER32	32 members in a page.

36.5.19 EBI_NANDCTRL - NAND Control 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				0
Access																											RW					RW
Name																											BANKSEL					EN

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	BANKSEL	0x0	RW	**NAND Flash Bank**
	This field sets the Memory Bank which is connected to a NAND Flash device			

Value	Mode	Description
0	BANK0	Memory bank 0 is connected to a NAND Flash device.
1	BANK1	Memory bank 1 is connected to a NAND Flash device.
2	BANK2	Memory bank 2 is connected to a NAND Flash device.
3	BANK3	Memory bank 3 is connected to a NAND Flash device.

3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	EN	0	RW	**NAND Flash Control Enable**
	This field enables NAND Flash control for the memory bank defined in BANK.			

36.5.20 EBI_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																														0	0	0
Access																														W1	W1	W1
Name																														ECCCLEAR	ECCSTOP	ECCSTART

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	ECCCLEAR	0	W1	**Error Correction Code Clear**
	Write to 1 to clear ECCPARITY.			
1	ECCSTOP	0	W1	**Error Correction Code Generation Stop**
	Write to 1 to stop ECC generation.			
0	ECCSTART	0	W1	**Error Correction Code Generation Start**
	Write to 1 to start ECC generation.			

36.5.21 EBI_STATUS - 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																			0	0		0	0	0				0				0
Access																			R	R		R	R	R				R				R
Name																			TFTDDEMPTY	DDACT		TFTPIXELFULL	TFTPIXEL1EMPTY	TFTPIXEL0EMPTY				ECCACT				AHBACT

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	TFTDDEMPTY	0	R	**EBI_TFTDD Register is Empty**
	Indicates that EBI_TFTDD register is empty.			
12	DDACT	0	R	**EBI Busy With Direct Drive Transactions**
	Indicates that EBI is busy with Direct Drive Transactions.			
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	TFTPIXELFULL	0	R	**EBI_TFTPIXEL0 is Full**
	Indicates that EBI_TFTPIXEL is full.			
9	TFTPIXEL1EMPTY	0	R	**EBI_TFTPIXEL1 is Empty**
	Indicates that EBI_TFTPIXEL1 is empty.			
8	TFTPIXEL0EMPTY	0	R	**EBI_TFTPIXEL0 is Empty**
	Indicates that EBI_TFTPIXEL0 is empty.			
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	ECCACT	0	R	**EBI ECC Generation Active**
	Indicates that EBI is generating ECC.			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	AHBACT	0	R	**EBI Busy With AHB Transaction**
	Indicates that EBI is busy with an AHB Transaction.			

36.5.22 EBI_ECCPARITY - ECC Parity 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	0x00000000																															
Access	R																															
Name	ECCPARITY																															

Bit	Name	Reset	Access	Description
31:0	ECCPARITY	0x00000000	R	**ECC Parity Data**
	ECC Parity Data.			

36.5.23 EBI_TFTCTRL - TFT Control 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		0		0x0					0	0x0		0	0			0x0				0x0	
Access									RW		RW		RW		RW					RW	RW		RW	RW			RW				RW	
Name									ALIASBANK		BANKSEL		ALIASBANKEN		WIDTH					COLOR1SRC	INTERLEAVE		FBCTRIG	SHIFTDCLKEN			MASKBLEND				DD	

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	ALIASBANK	0x0	RW	**Graphic Bank Select Aliasing**

This field sets the Memory Bank Select Alaising to enable Frame Buffer and Data sharing a common Memory Bank.

Value	Mode	Description
0	ALIASBANK0	Graphic Bank Select is alias to Bank Select 0
1	ALIASBANK1	Graphic Bank Select is alias to Bank Select 1
2	ALIASBANK2	Graphic Bank Select is alias to Bank Select 2
3	ALIASBANK3	Graphic Bank Select is alias to Bank Select 3

21:20	BANKSEL	0x0	RW	**Graphics Bank**

This field sets the Memory Bank containing the Frame Buffer

Value	Mode	Description
0	BANK0	Memory bank 0 is used for Direct Drive, Masking, and Alpha Blending.
1	BANK1	Memory bank 1 is used for Direct Drive, Masking, and Alpha Blending.
2	BANK2	Memory bank 2 is used for Direct Drive, Masking, and Alpha Blending.
3	BANK3	Memory bank 3 is used for Direct Drive, Masking, and Alpha Blending.

19	ALIASBANKEN	0	RW	**Alias to Graphics Bank Enable**

When set, Alias Bank Select is mapped to Graphic Bank Select.

Value	Mode	Description
0	DISABLE	Alias bank is disabled.
1	ENABLE	Alias bank is enabled.

18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	WIDTH	0x0	RW	**TFT Transaction Width**

This field sets TFT tranaction width.

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	BYTE		TFT Data is 8 bit wide.
	1	HALFWORD		TFT Data is 16 bit wide.
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	COLOR1SRC	0	RW	**Masking/Alpha Blending Color1 Source**
	This field sets the Masking/Alpha Blending Color1 Source.			
	Value	Mode		Description
	0	MEM		Masking/Alpha Blending color 1 is read from external memory.
	1	PIXEL1		Masking/Alpha Blending color 1 is read from EBI_TFTPIXEL1.
11:10	INTERLEAVE	0x0	RW	**Interleave Mode**
	This field sets the TFT Direct Drive Interleave mode.			
	Value	Mode		Description
	0	UNLIMITED		Allow unlimited interleaved EBI accesses per EBI_DCLK period. This can cause jitter on the EBI_DCLK
	1	ONEPERDCLK		Allow 1 interleaved EBI access per EBI_DCLK period.
	2	PORCH		Only allow EBI accesses during TFT porches.
9	FBCTRIG	0	RW	**TFT Frame Base Copy Trigger**
	Sets the trigger on which the TFTFRAMEBASE is copied into an internal buffer. Direct Drive address generation is based on the internal buffer.			
	Value	Mode		Description
	0	VSYNC		TFTFRAMEBASE is buffered on the vertical synchronization event EBI_VSYNC.
	1	HSYNC		TFTFRAMEBASE is buffered on the horizontal synchronization event EBI_HSYNC.
8	SHIFTDCLKEN	0	RW	**TFT EBI_DCLK Shift Enable**
	When this bit is set, EBI_DCLK edges are driven off the negative (instead of the positive) edge of the internal clock. SHIFTDCLKEN is only allowed to be set to 1 if TFTHOLD in EBI_TFTTIMING is at least 1.			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:2	MASKBLEND	0x0	RW	**TFT Mask and Blend Mode**
	This field sets the Mask and Blend Mode.			
	Value	Mode		Description
	0	DISABLED		Masking and Blending are disabled.
	1	IMASK		Internal Masking is enabled.
	2	IALPHA		Internal Alpha Blending is enabled.
	3	IMASKALPHA		Internal Masking and Alpha Blending are enabled.
	4	EFBMASK		External Frame Buffer Masking is enabled.

Bit	Name	Reset	Access	Description
	5	EFBALPHA		External Frame Buffer Alpha Blending is enabled.
	6	EFBMASKALPHA		External Frame Buffer Masking and Alpha Blending are enabled.
	7	IFBMASK		Internal Frame Buffer Masking is enabled.
	8	IFBALPHA		Internal Frame Buffer Alpha Blending is enabled.
	9	IFBMASKALPHA		Internal Frame Buffer Masking and Alpha Blending are enabled.
1:0	DD	0x0	RW	**TFT Direct Drive Mode**
	This field sets the Direct Mode.			

Value	Mode	Description
0	DISABLED	Direct Drive is disabled.
1	INTERNAL	Direct Drive from internal memory enabled and started.
2	EXTERNAL	Direct Drive from external memory enabled and started.

36.5.24 EBI_TFTSTATUS - TFT Status 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		0x0000																				0x000										
Access		R																				R										
Name		VCNT																				HCNT										

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	VCNT	0x0000	R	**Vertical Count**
	Contains the current line position within a frame (initial line in vertical back porch has VCNT = 0).			
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:0	HCNT	0x000	R	**Horizontal Count**
	Contains the current pixel position within a line (initial pixel in horizontal backporch has HCNT = 0).			

36.5.25 EBI_TFTCOLORFORMAT - Color Format 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	
Access																							RW								RW	
Name																							PIXEL1FORMAT								PIXEL0FORMAT	

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	PIXEL1FORMAT	0x0	RW	**Source and Destination Pixel Color Format**

Value	Mode	Description
0	RGB555	RGB data is 555
1	RGB565	RGB data is 565
2	RGB666	RGB data is 666
3	RGB888	RGB data is 888

Bit	Name	Reset	Access	Description
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	PIXEL0FORMAT	0x0	RW	**Sprite Pixel Color Format**

Value	Mode	Description
0	ARGB0555	ARGB data is 0555
1	ARGB0565	ARGB data is 0565
2	ARGB0666	ARGB data is 0666
3	ARGB0888	ARGB data is 0888
4	ARGB5555	ARGB data is 5555
5	ARGB6565	ARGB data is 6565
6	ARGB6666	ARGB data is 6666
7	ARGB8888	ARGB data is 8888

36.5.26 EBI_TFTFRAMEBASE - TFT Frame Base 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					0x0000000																											
Access					RW																											
Name					FRAMEBASE																											

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*		
27:0	FRAMEBASE	0x0000000	RW	**Frame Base Address**
	Sets the frame base address.			

36.5.27 EBI_TFTSTRIDE - TFT Stride 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																					0x000											
Access																					RW											
Name																					HSTRIDE											

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	HSTRIDE	0x000	RW	**Horizontal Stride**
	Sets the horizontal stride added to the Direct Drive address at the end of each line.			

36.5.28 EBI_TFTSIZE - TFT Size 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							0x000																0x000									
Access							RW																RW									
Name							VSZ																HSZ									

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*		
25:16	VSZ	0x000	RW	**Vertical Size (excluding Porches)**
	Sets the vertical size in lines. Set to required size minus 1.			
15:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9:0	HSZ	0x000	RW	**Horizontal Size (excluding Porches)**
	Sets the horizontal size in pixels. Set to required size minus 1.			

36.5.29 EBI_TFTHPORCH - TFT Horizontal Porch 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				0x00										0x00									0x00						
Access			RW				RW										RW									RW						
Name			HSYNCSTART				HBPORCH										HFPORCH									HSYNC						

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	HSYNCSTART	0x0	RW	**HSYNC Start Delay**
	Sets the HSYNC start position into the horizontal back porch in DCLK cycles.			
27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:18	HBPORCH	0x00	RW	**Horizontal Back Porch Size**
	Sets the horizontal back porch size in pixels.			
17:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	HFPORCH	0x00	RW	**Horizontal Front Porch Size**
	Sets the horizontal front porch size in pixels.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	HSYNC	0x00	RW	**Horizontal Synchronization Pulse Width**
	Sets the horizontal synchronization pulse width. Set to required width minus 1. Width is reduced in case HSYNCSTART > 0.			

36.5.30 EBI_TFTVPORCH - TFT Vertical Porch Register

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	0x000												0x000													0x00						
Access	RW												RW													RW						
Name	VBPORCH												VFPORCH													VSYNC						

Bit	Name	Reset	Access	Description
31:20	VBPORCH	0x000	RW	**Vertical Back Porch Size**
	Sets the Vertical back porch size in pixels.			
19:8	VFPORCH	0x000	RW	**Vertical Front Porch Size**
	Sets the Vertical front porch size in pixels.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	VSYNC	0x00	RW	**Vertical Synchronization Pulse Width**
	Sets the Vertical synchronization pulse width. Set to required width minus 1.			

36.5.31 EBI_TFTTIMING - TFT Timing 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				0x0			0x000												0x000											
Access		RW				RW			RW												RW											
Name		TFTHOLD				TFTSETUP			TFTSTART												DCLKPERIOD											

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	TFTHOLD	0x0	RW	**TFT Hold Time**
	Sets the number of internal clock cycles the RGB data is held after the active edge of EBI_DCLK.			
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26:24	TFTSETUP	0x0	RW	**TFT Setup Time**
	Sets the number of internal clock cycles the RGB data is driven before the active edge of EBI_DCLK.			
23:12	TFTSTART	0x000	RW	**TFT Direct Drive Transaction Start**
	Sets the starting position of the External Direct Drive Transaction relative to the DCLK inactive edge.			
11:0	DCLKPERIOD	0x000	RW	**TFT Direct Drive Transaction (EBI_DCLK) Period**
	Sets the Direct Drive transaction (EBI_DCLK) period in internal cycles. Set to required cycle count minus 1.			

36.5.32 EBI_TFTPOLARITY - TFT Polarity 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																												0	0	0	0	0
Access																												RW	RW	RW	RW	RW
Name																												VSYNCPOL	HSYNCPOL	DATAENPOL	DCLKPOL	CSPOL

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	VSYNCPOL	0	RW	**VSYNC Polarity**

Sets the polarity of the EBI_VSYNC line.

Value	Mode	Description
0	ACTIVELOW	VSYNC is active low.
1	ACTIVEHIGH	VSYNC is active high.

Bit	Name	Reset	Access	Description
3	HSYNCPOL	0	RW	**Address Latch Polarity**

Sets the polarity of the EBI_HSYNC line.

Value	Mode	Description
0	ACTIVELOW	HSYNC is active low.
1	ACTIVEHIGH	HSYNC is active high.

Bit	Name	Reset	Access	Description
2	DATAENPOL	0	RW	**TFT DATAEN Polarity**

Sets the polarity of the EBI_DATAEN line.

Value	Mode	Description
0	ACTIVELOW	DATAEN is active low.
1	ACTIVEHIGH	DATAEN is active high.

Bit	Name	Reset	Access	Description
1	DCLKPOL	0	RW	**TFT DCLK Polarity**

Sets the active edge polarity of the EBI_DCLK line.

Value	Mode	Description
0	ACTIVEFALLING	DCLK falling edge is the active edge.
1	ACTIVERISING	DCLK rising edge the active edge.

Bit	Name	Reset	Access	Description
0	CSPOL	0	RW	**TFT Chip Select Polarity**

Sets the polarity of the EBI_CSTFT line.

Value	Mode	Description
0	ACTIVELOW	CSTFT is active low.

Bit	Name	Reset	Access	Description
	1	ACTIVEHIGH		CSTFT is active high.

36.5.33 EBI_TFTDD - TFT Direct Drive Data Register

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									0x000000																							
Access									RW																							
Name									DATA																							

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	DATA	0x000000	RW	**TFT Direct Drive Data From Internal Memory**
	Sets the RGB value used when Direct Drive from internal memory is used (DD = INTERNAL)			

36.5.34 EBI_TFTALPHA - TFT Alpha Blending Register

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																												0x000				
Access																												RW				
Name																												ALPHA				

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	ALPHA	0x000	RW	**TFT Alpha Blending Factor**
	Sets the alpha blending factor. The maximum value is map to 255.			

36.5.35 EBI_TFTPIXEL0 - TFT Pixel 0 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									0x000000																					
Access									RW																					
Name									DATA																					

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	DATA	0x000000	RW	**RGB Data**
	Sets the RGB data value according to the format defined in RGBMODE.			

36.5.36 EBI_TFTPIXEL1 - TFT Pixel 1 Register

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									0x000000																					
Access									RW																					
Name									DATA																					

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	DATA	0x000000	RW	**RGB Data**
	Sets the RGB data value according to the format defined in RGBMODE.			

36.5.37 EBI_TFTPIXEL - TFT Alpha Blending Result Pixel Register

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									0x000000																					
Access									R																					
Name									DATA																					

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	DATA	0x000000	R	**Alpha Blending Result**
	RGB result of Alpha Blending operation according to the format defined in RGBMODE.			

36.5.38 EBI_TFTMASK - TFT Masking Register

Offset	Bit Position																													
0x09C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x000000																					
Access									RW																					
Name									TFTMASK																					

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	TFTMASK	0x000000	RW	**TFT Mask Value**
	Sets the mask value. Data write transactions matching this value are suppressed.			

36.5.39 EBI_IF - Interrupt Flag Register

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																							0	0	0	0	0	0	0	0	0	0
Access																							R	R	R	R	R	R	R	R	R	R
Name																							TFTPIXELOF	TFTPIXELFULL	TFTPIXEL1EMPTY	TFTPIXEL0EMPTY	DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC

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	TFTPIXELOF	0	R	**EBI_TFTPIXEL Register Overflow Interrupt Flag**
	Set when TFTPIXEL register data is over written before it is read.			
8	TFTPIXELFULL	0	R	**EBI_TFTPIXEL is Full Interrupt Flag**
	Set when Alpha Blending result TFTPIXEL register is fulled.			
7	TFTPIXEL1EMPTY	0	R	**EBI_TFTPIXEL1 is Empty Interrupt Flag**
	Set when in Internal Alpha Blending mode TFTPIXEL1 register has been processed.			
6	TFTPIXEL0EMPTY	0	R	**EBI_TFTPIXEL0 is Empty Interrupt Flag**
	Set when in Internal Alpha Blending mode TFTPIXEL0 register has been processed.			
5	DDJIT	0	R	**Direct Drive Jitter Interrupt Flag**
	Set when DCLKPERIOD is not met.			
4	DDEMPTY	0	R	**Direct Drive Data Empty Interrupt Flag**
	Set when Direct Drive engine EBI_TFTDD data is empty.			
3	VFPORCH	0	R	**Vertical Front Porch Interrupt Flag**
	Set at beginning of Vertical Front Porch.			
2	VBPORCH	0	R	**Vertical Back Porch Interrupt Flag**
	Set at end of Vertical Back Porch.			
1	HSYNC	0	R	**Horizontal Sync Interrupt Flag**
	Set at Horizontal Sync pulse.			
0	VSYNC	0	R	**Vertical Sync Interrupt Flag**
	Set at Vertical Sync pulse.			

36.5.40 EBI_IFS - Interrupt Flag Set Register

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																							0	0	0	0	0	0	0	0	0	0
Access																							W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																							TFTPIXELOF	TFTPIXELFULL	TFTPIXEL1EMPTY	TFTPIXEL0EMPTY	DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC

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	TFTPIXELOF	0	W1	**EBI_TFTPIXEL Overflow Interrupt Flag Set**
	Write to 1 to set EBI_TFTPIXEL Overflow Interrupt flag.			
8	TFTPIXELFULL	0	W1	**EBI_TFTPIXEL Full Interrupt Flag Set**
	Write to 1 to set EBI_TFTPIXEL Full Interrupt flag.			
7	TFTPIXEL1EMPTY	0	W1	**EBI_TFTPIXEL1 Empty Interrupt Flag Set**
	Write to 1 to set EBI_TFTPIXEL1 Empty Interrupt flag.			
6	TFTPIXEL0EMPTY	0	W1	**EBI_TFTPIXEL0 Empty Interrupt Flag Set**
	Write to 1 to set EBI_TFTPIXEL0 Empty Interrupt flag.			
5	DDJIT	0	W1	**Direct Drive Jitter Interrupt Flag Set**
	Write to 1 to set Direct Drive Jitter Interrupt flag.			
4	DDEMPTY	0	W1	**Direct Drive Data Empty Interrupt Flag Set**
	Write to 1 to set Direct Drive Data Empty Interrupt flag.			
3	VFPORCH	0	W1	**Vertical Front Porch Interrupt Flag Set**
	Write to 1 to set Vertical Front Porch Interrupt flag.			
2	VBPORCH	0	W1	**Vertical Back Porch Interrupt Flag Set**
	Write to 1 to set Vertical Back Porch Interrupt flag.			
1	HSYNC	0	W1	**Horizontal Sync Interrupt Flag Set**
	Write to 1 to set Horizontal Sync interrupt flag.			
0	VSYNC	0	W1	**Vertical Sync Interrupt Flag Set**
	Write to 1 to set Vertical Sync interrupt flag.			

36.5.41 EBI_IFC - Interrupt Flag Clear Register

Offset	Bit Position																											
0x0A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0	0	0	0	0	0	0	0	0	0
Access																							(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name																							TFTPIXELOF	TFTPIXELFULL	TFTPIXEL1EMPTY	TFTPIXEL0EMPTY	DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC

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	TFTPIXELOF	0	(R)W1	**EBI_TFTPIXEL Overflow Interrupt Flag Clear**
	Write to 1 to clear EBI_TFTPIXEL Overflow Interrupt flag.			
8	TFTPIXELFULL	0	(R)W1	**EBI_TFTPIXEL Full Interrupt Flag Clear**
	Write to 1 to clear EBI_TFTPIXEL Full Interrupt flag.			
7	TFTPIXEL1EMPTY	0	(R)W1	**EBI_TFTPIXEL1 Empty Interrupt Flag Clear**
	Write to 1 to clear EBI_TFTPIXEL1 Empty Interrupt flag.			
6	TFTPIXEL0EMPTY	0	(R)W1	**EBI_TFTPIXEL0 Empty Interrupt Flag Clear**
	Write to 1 to clear EBI_TFTPIXEL0 Empty Interrupt flag.			
5	DDJIT	0	(R)W1	**Direct Drive Jitter Interrupt Flag Clear**
	Write to 1 to clear Direct Drive Jitter Interrupt flag.			
4	DDEMPTY	0	(R)W1	**Direct Drive Data Empty Interrupt Flag Clear**
	Write to 1 to clear Direct Drive Data Empty Interrupt flag.			
3	VFPORCH	0	(R)W1	**Vertical Front Porch Interrupt Flag Clear**
	Write to 1 to clear Vertical Front Porch interrupt flag.			
2	VBPORCH	0	(R)W1	**Vertical Back Porch Interrupt Flag Clear**
	Write to 1 to clear Vertical Back Porch interrupt flag.			
1	HSYNC	0	(R)W1	**Horizontal Sync Interrupt Flag Clear**
	Write to 1 to clear Horizontal Sync interrupt flag.			
0	VSYNC	0	(R)W1	**Vertical Sync Interrupt Flag Clear**
	Write to 1 to clear Vertical Sync interrupt flag.			

36.5.42 EBI_IEN - Interrupt Enable Register

Offset	Bit Position																											
0x0AC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0	0	0	0	0	0	0	0	0	0
Access																							RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																							TFTPIXELOF	TFTPIXELFULL	TFTPIXEL1EMPTY	TFTPIXEL0EMPTY	DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC

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	TFTPIXELOF	0	RW	**EBI_TFTPIXEL Overflow Interrupt Enable**
	Set to enable interrupt on EBI_TFTPIXEL Full Interrupt flag.			
8	TFTPIXELFULL	0	RW	**EBI_TFTPIXEL Full Interrupt Enable**
	Set to enable interrupt on EBI_TFTPIXEL Full Interrupt flag.			
7	TFTPIXEL1EMPTY	0	RW	**EBI_TFTPIXEL1 Empty Interrupt Enable**
	Set to enable interrupt on EBI_TFTPIXEL1 Empty Interrupt flag.			
6	TFTPIXEL0EMPTY	0	RW	**EBI_TFTPIXEL0 Empty Interrupt Enable**
	Set to enable interrupt on EBI_TFTPIXEL0 Empty Interrupt flag.			
5	DDJIT	0	RW	**Direct Drive Jitter Interrupt Enable**
	Set to enable interrupt on Direct Drive Jitter Interrupt flag.			
4	DDEMPTY	0	RW	**Direct Drive Data Empty Interrupt Enable**
	Set to enable interrupt on Direct Drive Data Empty Interrupt flag.			
3	VFPORCH	0	RW	**Vertical Front Porch Interrupt Enable**
	Set to enable interrupt on beginning of Vertical Front Porch interrupt flag.			
2	VBPORCH	0	RW	**Vertical Back Porch Interrupt Enable**
	Set to enable interrupt on end of Vertical Back Porch interrupt flag.			
1	HSYNC	0	RW	**Horizontal Sync Interrupt Enable**
	Set to enable interrupt on Horizontal Sync interrupt flag.			
0	VSYNC	0	RW	**Vertical Sync Interrupt Enable**
	Set to enable interrupt on Vertical Sync interrupt flag.			

36.5.43 EBI_ROUTEPEN - I/O Routing Register

Offset	Bit Position																													
0x0B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset						0	0	0		0x00					0x0					0					0	0	0	0	0	0	0	0
Access						RW	RW	RW		RW					RW					RW					RW	RW	RW	RW	RW	RW	RW	RW
Name						CSTFTPEN	DATAENPEN	TFTPEN		APEN					ALB					NANDPEN					BLPEN	ARDYPEN	ALEPEN	CS3PEN	CS2PEN	CS1PEN	CS0PEN	EBIPEN

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	CSTFTPEN	0	RW	**EBI_CSTFT Pin Enable**
	When set, the EBI_CSTFT pin is enabled			
25	DATAENPEN	0	RW	**EBI_DATA Pin Enable**
	When set, the EBI_DATAEN pin is enabled			
24	TFTPEN	0	RW	**EBI_TFT Pin Enable**
	When set, the EBI_DCLK, EBI_VSYNC and EBI_HSYNC pins are enabled			
23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:18	APEN	0x00	RW	**EBI_A Pin Enable**
	Selects which non-multiplexed address lines are enabled on EBI_A. The lower bound L is set to 0, 8, 16 or 24 as defined in the ALB field.			

Value	Mode	Description
0	A0	All EBI_A pins are disabled.
5	A5	EBI_A[4:L] pins enabled.
6	A6	EBI_A[5:L] pins enabled.
7	A7	EBI_A[6:L] pins enabled.
8	A8	EBI_A[7:L] pins enabled.
9	A9	EBI_A[8:L] pins enabled.
10	A10	EBI_A[9:L] pins enabled.
11	A11	EBI_A[10:L] pins enabled.
12	A12	EBI_A[11:L] pins enabled.
13	A13	EBI_A[12:L] pins enabled.
14	A14	EBI_A[13:L] pins enabled.
15	A15	EBI_A[14:L] pins enabled.
16	A16	EBI_A[15:L] pins enabled.
17	A17	EBI_A[16:L] pins enabled.

Bit	Name	Reset	Access	Description
	18	A18		EBI_A[17:L] pins enabled.
	19	A19		EBI_A[18:L] pins enabled.
	20	A20		EBI_A[19:L] pins enabled.
	21	A21		EBI_A[20:L] pins enabled.
	22	A22		EBI_A[21:L] pins enabled.
	23	A23		EBI_A[22:L] pins enabled.
	24	A24		EBI_A[23:L] pins enabled.
	25	A25		EBI_A[24:L] pins enabled.
	26	A26		EBI_A[25:L] pins enabled.
	27	A27		EBI_A[26:L] pins enabled.
	28	A28		EBI_A[27:L] pins enabled.
17:16	ALB	0x0	RW	**Sets the Lower Bound for EBI_A Enabling**
	Sets the lower bound of the EBI_A lines which can be enabled in the APEN field.			

Value	Mode	Description
0	A0	Address lines from EBI_A[0] and upwards can be enabled via APEN.
1	A8	Address lines from EBI_A[8] and upwards can be enabled via APEN.
2	A16	Address lines from EBI_A[16] and upwards can be enabled via APEN.
3	A24	Address lines from EBI_A[24] and upwards can be enabled via APEN.

15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	NANDPEN	0	RW	**NANDRE and NANDWE Pin Enable**
	When set, the NANDREn and NANDWEn Pin pins are enabled			
11:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	BLPEN	0	RW	**EBI_BL[1:0] Pin Enable**
	When set, the EBI_BL[1:0] pins are enabled			
6	ARDYPEN	0	RW	**EBI_ARDY Pin Enable**
	When set, the EBI_ARDY pin is enabled			
5	ALEPEN	0	RW	**EBI_ALE Pin Enable**
	When set, the EBI_ALE pin is enabled			
4	CS3PEN	0	RW	**EBI_CS3 Pin Enable**
	When set, the EBI_CS3 pin is enabled			
3	CS2PEN	0	RW	**EBI_CS2 Pin Enable**
	When set, the EBI_CS2 pin is enabled			
2	CS1PEN	0	RW	**EBI_CS1 Pin Enable**
	When set, the EBI_CS1 pin is enabled			

Bit	Name	Reset	Access	Description
1	CS0PEN	0	RW	**EBI_CS0 Pin Enable**
	When set, the EBI_CS0 pin is enabled			
0	EBIPEN	0	RW	**EBI Pin Enable**
	When set, the EBI_AD[15:0], EBI_WEn and EBI_REn pins are enabled			

36.5.44 EBI_ROUTELOC0 - I/O Routing Location Register

Offset	Bit Position																													
0x0B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			TFTLOC								NANDLOC								CSLOC								EBILOC					

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	TFTLOC	0x00	RW	**I/O Location**

Decides the location of the EBI_DCLK, EBI_VSYNC, EBI_HSYNC, EBI_DATAEN and CSTFT pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 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*		
21:16	NANDLOC	0x00	RW	**I/O Location**

Decides the location of the NANDREn and NANDWEn pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

Bit	Name	Reset	Access	Description
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CSLOC	0x00	RW	**I/O Location**

Decides the location of the EBI_CS pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

Bit	Name	Reset	Access	Description
	3	LOC3		Location 3
	4	LOC4		Location 4
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	EBILOC	0x00	RW	**I/O Location**

Decides the location of the EBI_WEn, EBI_REn, EBI_BL and EBI_ALE pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

36.5.45 EBI_ROUTELOC1 - I/O Routing Location Register

Offset	Bit Position																													
0x0B8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x00								0x00								0x00					
Access											RW								RW								RW					
Name											RDYLOC								ALOC								ADLOC					

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	RDYLOC	0x00	RW	**I/O Location**

Decides the location of the EBI_ARDY pin.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5

15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	ALOC	0x00	RW	**I/O Location**

Decides the location of the EBI_A pin.

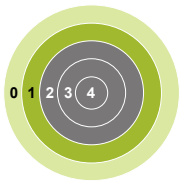
Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

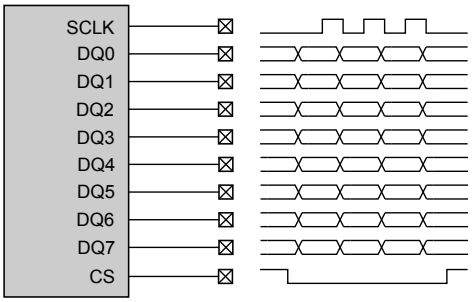
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	ADLOC	0x00	RW	**I/O Location**

Decides the location of the EBI_AD pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1

37. QSPI - Quad- and Octal-SPI Flash Controller





Quick Facts

What?

QSPI provides large bandwidth and low latency to external flash memory for data storage and code execution with a low pin count.

Why?

External flash allows larger code footprint, in-field upgrades, data logging, etc.

How?

The high clock speed, low pin count QSPI controller gives high throughput while keeping I/O usage minimal. Memory mapped flash interface allows random access and code execution. Internal instruction cache reduces latency.

37.1 Introduction

The QSPI provides access to a wide range of flash devices with wide I/O busses. The I/O and clocking configuration is flexible and supports many types of devices. Up to 8-bit wide interfaces are supported. The QSPI handles opcodes, status flag polling and timing configuration automatically.

The external flash is mapped directly to internal memory to allow random access to any word in the flash and direct code execution. The instruction cache minimizes latency and allows efficient code execution. Execute in Place (XIP) is supported for devices with this feature.

Large data chunks can be transferred with DMA as efficiently as possible with high throughput and minimimal bus load, utilizing an integrated 1 kB SRAM FIFO.

37.2 Features

- Memory mapped ‘direct’ mode of operation for performing flash data transfers and executing code from flash memory
- Software triggered ‘indirect’ mode of operation for performing low latency and non processor intensive flash data transfers
- Support for XIP (Execute in Place), sometimes referred to as continuous mode
- Support for DDR Mode (Double Data Rate)
- Support for single, dual, quad or octal transfers
- Programmable device sizes
- Programmable write protected regions to block system writes from taking effect
- Programmable delays between transactions
- Up to 2 external device selects
- Software Triggered Instruction Generator (STIG) allowing software to generate any flash command, including data transfers up to 8-bytes at a time

37.3 Functional Description

37.3.1 Direct Access Controller

The Direct Access Controller (DAC) implements a direct, memory-mapped interface to the external flash. When using the DAC, a read or write to the QSPI memory region directly triggers a read or write to the flash memory. For reads the DAC will automatically send one additional read request to the flash. This predicted read helps to reduce the effective latency for sequential reads.

Once a page program cycle has been started, the QSPI will automatically poll for the write cycle to complete before allowing any further read/write access to complete. This is achieved by stalling any subsequent memory access to the QSPI until the operation has completed. The QSPI will automatically detect page boundaries and automatically initiate polling to ensure the page is ready to accept new data, provided the correct page size is configured in QSPIn_DEVSIZECONFIG.

Note: The page boundary detection is only guaranteed for word (4-byte) aligned accesses. It is therefore recommended to ensure that all writes are word aligned.

37.3.2 Indirect Access Controller

The Indirect Access Controller (INDAC) allows high throughput data transfer with as little system overhead as possible. Using this mode, software controls the transfers using several registers. An integrated 1 kB SRAM FIFO allows buffering when the QSPI is writing to or reading from the external memory. Interrupts and fill-level registers help guide when there is space in or data available in the SRAM. All transfers (reads and writes) with the INDAC should be 32-bits wide and word-aligned.

37.3.2.1 Indirect Trigger Range

An indirect trigger range must be defined before initiating an indirect transfer. The QSPIn_INDAHBADDRTRIGGER defines the start address for the trigger range. The QSPIn_INDIRECTTRIGGERADDRRANGE register defines the address range. Given the start address in INDAHBADDRTRIGGER and the address range in INDIRECTTRIGGERADDRRANGE, then any access to address ADDR is treated as an indirect access according to Figure 37.1 Indirect Trigger Range on page 1392.

$$INDAHBADDRTRIGGER <= ADDR < INDAHBADDRTRIGGER + (2^{INDIRECTTRIGGERADDRRANGE})$$

Figure 37.1. Indirect Trigger Range

Any access that is not treated as indirect will be treated as a direct access and handled by the Direct Access Controller.

The trigger address is decoupled from the flash address. Any access within the trigger range will be treated as an indirect transfer (read or write). Software can read (or write) from the same address repeatedly or decide to increase the address for each transfer (as long as the address is within the trigger range). The actual address sent to the external flash is controlled by the QSPIn_INDIRECTREADXFERSTART and QSPIn_INDIRECTWRITEXFERSTART registers.

37.3.2.2 SRAM FIFO

The FIFO consists of a 1 kB SRAM and an additional 32-bit wide register resulting in a total of 257 32-bit locations. The FIFO is used for both reading and writing and is segmented into two parts. The lower part used for reads, while the upper part is used for writes. The partitioning is programmable via the QSPIn_SRAMPARTITIONCFG. This allows the system to decide which part should be largest. The size (in 32-bit words) of each part of the FIFO can be calculated with the following expressions where P is the value programmed in QSPIn_SRAMPARTITIONCFG:

$$n_{READ} = P + 1$$

Figure 37.2. Read FIFO Depth (words)

$$n_{WRITE} = 256 - P$$

Figure 37.3. Write FIFO Depth (words)

E.g. if QSPIn_SRAMPARTITIONCFG is set to 2 then the read FIFO has room for 3 words and the write fifo is 254 words. Note that software should never program 0xFF or 0x00 in QSPIn_SRAMPARTITIONCFG, i.e. it is not allowed to program either part to zero.

37.3.2.3 Indirect Read

To start an indirect read, software must configure how much data is required and starting from what address. The start address and total number of bytes to be fetched is defined in QSPIn_INDIRECTREADXFERSTART and QSPIn_INDIRECTREADXFERNUMBYTES respectively. Setting the START bit in QSPIn_INDIRECTREADXFERCTRL starts an indirect read operation, and the RDSTATUS will be available to check the status. The address sent to the flash is increased automatically until the given number of bytes have all been transferred. The INDAC will start filling the SRAM FIFO, but will not overwrite the contents if the FIFO is full. The controller will pause reading from flash until there is space available in the FIFO.

Software can access the SRAM FIFO fill level directly via the QSPIn_SRAMFILL and then decide when the data should be fetched from the FIFO. The QSPIn_INDIRECTREADXFERWATERMARK register can be used to set up a watermark. When the fill level passes this

watermark, an interrupt is generated. If the watermark value is larger than zero, the watermark interrupt is also generated when the final byte of data has been read by the QSPI controller and placed in the SRAM even if the actual SRAM fill level has not risen above the watermark. This feature is useful to avoid software tracking how much data has been read and resetting the watermark value for the last few bytes of an indirect read transfer. An indirect operation may be cancelled at any time by setting the CANCEL bit in QSPIn_INDIRECTREADXFERCTRL.

37.3.2.4 Indirect Read Process

To read data with the indirect controller, the following sequence can be followed:

1. Setup the indirect transfer's flash start address in QSPIn_INDIRECTREADXFERSTART
2. Setup the number of bytes to be transferred in QSPIn_INDIRECTREADXFERNUMBYTES
3. Setup the indirect transfer's trigger region in QSPIn_INDAHBADDRTRIGGER and QSPIn_INDIRECTTRIGGERADDRRANGE. This is the AHB address region that is used for indirect trasfers and must reside in one of the two QSPI memory spaces starting at 0x04000000 (code space) and 0xC0000000 (system space).
4. Configure the QSPIn_INDIRECTREADXFERWATERMARK (optional)
5. Trigger Indirect Read access by setting START bit in QSPIn_INDIRECTREADXFERCTRL
6. If the watermark interrupt feature is used, wait for watermark interrupt. Otherwise poll the SRAM fill level to decide when sufficient data is in the SRAM to trigger data fetches.
7. Read the expected amount of data from trigger region (data will be fetched from SRAM FIFO). If there is still more data to fetch in order to complete the indirect read transfer, then loop back to 6.

37.3.2.5 Indirect Write Controller

To start an indirect write, software must configure how much data is required and starting from what address. The start address and total number of bytes to be fetched is defined in QSPIn_INDIRECTWRITEXFERSTART and QSPIn_INDIRECTWRITEXFERNUMBYTES respectively. Setting the START bit in QSPIn_INDIRECTWRITEXFERCTRL starts an indirect write operation, and the WRSTATUS will be available to check the status. The address sent to the flash is increased automatically until the given number of bytes have all been transferred. If software tries to write while the FIFO is full, the QSPI will stall the bus until there is space in the FIFO. It is recommended to always make sure there is enough space in the FIFO before writing new data. Failing to do so can cause the write to stall for a considerable amount of time.

Software can access the SRAM FIFO fill level directly via the QSPIn_SRAMFILL and then decide when it should write new data. The QSPIn_INDIRECTWRITEXFERWATERMARK register can be used to set up a watermark. When the fill level drops below this watermark, an interrupt is generated. When the SRAM holds a number of bytes equal to or greater than the size of a flash page (which itself is programmable in the QSPIn_DEVSIZECONFIG register) or when the SRAM holds all remaining bytes of the currently executing indirect transfer, the QSPI will initiate a write to the flash. An indirect operation may be cancelled at any time by setting the CANCEL bit in QSPIn_INDIRECTWRITEXFERCTRL.

37.3.2.6 Indirect Write Process

When writing data with the Indirect Write Controller, the following sequence can be followed:

1. Configure the indirect transfer's flash start address in QSPIn_INDIRECTWRITEXFERSTART
2. Configure the number of bytes to be transferred in QSPIn_INDIRECTWRITEXFERNUMBYTES
3. Setup the indirect transfer's trigger region in QSPIn_INDAHBADDRTRIGGER and QSPIn_INDIRECTTRIGGERADDRRANGE. This is the AHB address region that is used for indirect trasfers and must reside in one of the two QSPI memory spaces starting at 0x04000000 (code space) and 0xC0000000 (system space).
4. Configure QSPIn_INDIRECTWRITEXFERWATERMARK (optional)
5. Trigger Indirect Write access by setting the START bit in QSPIn_INDIRECTWRITEXFERCTRL
6. Write one full flash page to trigger region at a time (data will be written to SRAM FIFO). Use the watermark interrupt or the SRAM fill level to determine when to write more data.
7. When all data have been written wait for the INOPSDONESTATUS bit in QSPIn_INDIRECTWRITEXFERCTRL or the INDIRECTOPDONE interrupt flag.

37.3.2.7 Indirect Access Queueing

Software is permitted to queue up to two indirect transfers for both write and read operations. Supporting two indirect operations allows a short turnaround time between the completion of one indirect operation and the start of the second. An INDIRECTREADREJECT interrupt is generated if an indirect operation was requested but could not be accepted because 2 indirect operations have already been queued by the QSPI.

Indirect access queuing is achieved by triggering the START bit of the indirect transfer control register (QSPIn_INDIRECTREADXFERCTRL or QSPIn_INDIRECTWRITEXFERCTRL) a second time before the first transfer is complete. The corresponding start address and number of bytes to transfer must be set up before the START bit is triggered for each transfer.

It is permitted for software to trigger an indirect read operation while an indirect write operation is in progress. Similarly it is permitted to trigger an indirect write while an indirect read operation is in progress. Indirect write operations will take overall precedence.

37.3.3 Software Triggered Instruction Generator

The direct and indirect access controllers are used to transfer data. In order to access the volatile and non-volatile configuration registers, the legacy SPI Status register, other status/protection registers as well as to perform erase functions, a separate software controller is required. The software triggered instruction generator, or STIG, is controlled using the QSPIn_FLASHCMDCTRL by setting up the command to issue to the flash device. This is a generic controller and can be used to perform any instruction that the flash device supports from the extended SPI protocol.

The CMDOPCODE field sets the opcode for the request. The CMDEXEC bit is used to trigger the command. The CMDEXECSTATUS bit can be used by software to poll the status of the command execution.

The opcode is always sent first. If there is an address to send, then the address (the size of which is also programmed in the same register) is sent next. The address itself is stored in QSPIn_FLASHCMDADDR. If Mode bits are enabled (ENBMODEBIT), then the MODE field in QSPIn_MODEBITCONFIG is sent right after the address.

If ENBMODEBIT in QSPIn_FLASHCMDCTRL and CRCENABLE in QSPIn_CONFIG are both enabled, STIG will replace XIP Mode bits with automatically calculated address CRC byte (needed for opcodes which require a CRC of the address).

If the NUMDUMMYCYCLES field is non-zero, the programmed number of dummy cycles are sent next.

In the case of a read, the number of bytes to read is programmed in NUMRDDATABYTES and the data is stored in QSPIn_FLASHRDDATALOWER and QSPIn_FLASHRDDATAUPPER after the STIG request completes.

In the case of a write, the number of bytes to write are programmed in NUMWRDATABYTES and the data should be written to QSPIn_FLASHWRDATALOWER and QSPIn_FLASHWRDATAUPPER before executing the STIG request.

A STIGREQINT interrupt is generated on the completion of a STIG operation. The occurrence of the interrupt indicates that the controller is ready to accept a new STIG request. It is important to notice that completion of the STIG request is not equivalent to completion it on SPI side. E.g. if STIG is performing a write operation the data is taken from the corresponding STIG register fields and put into TX FIFO. Since all bytes to write are known, another STIG can be queued before SPI traffic is complete.

There are some commands which require more data to read than 8 bytes (for example READ ID command). The STIG Memory Bank can be used to accommodate extra data if needed. The STIG Memory Bank is controlled by the STIGMEMBANKEN bit in QSPIn_FLASHCMDCTRL. If enabled, an extra 8 bytes (16 total) can be read using STIG. If the number of bytes to read exceeds the Memory Bank depth, remaining data will overwrite the STIG Memory Bank locations starting from its first address.

The STIG read data registers (QSPIn_FLASHRDDATALOWER and QSPIn_FLASHRDDATAUPPER) keep the last 8 bytes read from the flash device by STIG when Memory Bank is enabled. In order to access more data, STIG Memory Bank data request should be triggered, controlled by QSPIn_FLASHCOMMANDCTRLMEM. The TRIGGERMEMBANKREQ bit is used to trigger the command, MEMBANKREQINPROGRESS used by software to poll the status of the command execution. When MEMBANKREQINPROGRESS toggles from 1 to 0, the byte of data (MEMBANKREADDATA) from corresponding address (MEMBANKADDR) is valid.

37.3.4 Arbitration

When multiple controllers are active simultaneously, a simple fixed-priority arbitration scheme is used to arbitrate between each interface and access the external flash. The fixed priority is defined as follows, highest priority first:

1. Indirect Access Write
2. Direct Access Write
3. STIG
4. Direct Access Read
5. Indirect Access Read

37.3.5 Memory Space

The QSPI is mapped to two different memory regions, one in code space (0x04000000 - 0x0BFFFFFF) and one in system space (0xC0000000 - 0xCFFFFFFF). See 4.2 Functional Description for details. Each region can be used to both write to and read from external flash. The system space region is accessible by both the CPU and DMA. The code space region is accessible by the CPU only. It is recommended to use the code space region for code execution and the system space region if the primary purpose is data storage.

Instruction fetches from the code space region are cached by the system instruction cache to achieve efficient code execution. The caching of instructions from QSPI can be disabled with the QSPICDIS bit in MSC_READCTRL. See 6.3.10 Instruction Cache for more information about the instruction cache.

37.3.5.1 External Flash Address

For INDAC or STIG based transfers the flash address is configured directly in QSPIn_INDIRECTREADXFERSTART, QSPIn_INDIRECTWRITEXFERSTART and QSPIn_FLASHCMDADDR.

For DAC the flash address is based on the AHB address. When ENBAHBADDRREMAP in QSPIn_CONFIG is disabled, flash address is used directly. When ENBAHBADDRREMAP is enabled, QSPIn_REMAPADDR is added to the AHB address. If the device is using 3 byte address, only the 3 lower byte of the internal address is sent to the flash.

DAC may also make use of the QSPI Address Decoder that automatically accesses the correct flash device based on memory address when multiple flash devices are connected. If the address decoder is *disabled*, the active flash device is controlled by the PERIPHCSLINES field in QSPIn_CONFIG. When the ENABLEAHBDECODER bit in QSPIn_CONFIG is set, the address decoder is *enabled* and the devices are combined to form one contiguous memory area. Boundaries of addresses are calculated based on configuration from QSPIn_DEVSIZECONFIG. Which device is accessed is determined automatically based on the internal address.

Each flash device should be ready to accept read commands (i.e. done with erase/write commands). To ensure the devices are ready, it is recommended to first do reads separately for each device without using the decoder.

The QSPI controller does not implement any specific address decoding to error incoming addresses that may lie outside the connected flash memory space.

37.3.6 Write Protection

In order to protect the flash device, a write protection feature is implemented in hardware and controlled from software. Any write detected (by using DAC controller), pointing to an area of flash that is protected, will generate a bus fault.

A programmable region of the flash device, defined as a number of flash "blocks" starting from a particular block number can be protected. Three programmable registers (QSPIn_LOWERWRPROT, QSPIn_UPPERWRPROT, QSPIn_WRPROTCTRL) are provided. The first register defines the flash block that is located at the bottom of the region to be protected. The second register defines the flash block that is located at the top of the region to be protected. Write protection is enabled by setting the ENB field in QSPIn_WRPROTCTRL. Setting the INV bit allows software to invert the region that is being protected, causing the programmed region to become the only areas of flash memory that is not protected from writes.

A block can be between 1 and 65K bytes, programmed via the QSPIn_DEVSIZECONFIG register.

37.3.7 SPI Command Translation

For writes, the write enable latch (WEL) within the flash device itself must be set before a write sequence can be issued. The QSPI will automatically issue the write enable (WREN) command before triggering a write command via DAC or INDAC. This feature can be turned off by setting the WELDIS bit in QSPIn_DEVINSTRWRCONFIG. The opcode for WREN is typically 0x06 and is common between devices.

During write operations, if another request comes in, the QSPI will wait until the device is ready before performing the request. This is achieved by sending the read status register (RDSR) opcode and checking the busy bit.

The WREN and RDSR opcodes are the only instructions that are sent to the flash under the hood. Any other special instructions to the flash (e.g. unprotect, erase commands) must be issued by the STIG.

37.3.8 SPI Transfer Configuration

In order to send the correct read and write opcodes, software should initialize the QSPIn_DEVINSTRRDCONFIG and QSPIn_DEVINSTRWRCONFIG registers. These registers include fields to configure the required instruction opcodes that is intended to be used to access the flash as well as instruction type (single, dual, quad or octal) and dummy cycles. The selection between SDR and DDR is done by configuring ENABLEDTRPROTOCOL bit in QSPIn_CONFIG. Setting this bit will make all SPI tranfers use DDR. If the read opcode configured by RDOPCODENONXIP in QSPIn_DEVINSTRRDCONFIG requires DDR only on address and data part of transfer, this must be configured by setting DDREN high while keeping ENABLEDTRPROTOCOL bit low. In this case all communication will be SDR, expecept for address and data phase of data read instructions.

The instruction type (INSTRTYPE) field is only included in QSPIn_DEVINSTRRDCONFIG, (it is not included in QSPIn_DEVINSTRWRCONFIG) as it is controlling instruction type for both reads and writes. If software sets this to anything other than 0, then the address transfer type and the data transfer type fields of both QSPIn_DEVINSTRRDCONFIG and QSPIn_DEVINSTRWRCONFIG are ignored. In this case the same mode (from INSTRTYPE) is used for all phases (opcode, address and data).

37.3.9 I/O Timing

Note that this chapter is not valid if the non-PHY mode is selected. Please refer to 37.3.13 Non-PHY mode for information about the non-PHY mode.

QSPInCLK is selected from a set of possible oscillators by controlling QSPInCLKSEL in CMU_QSPICTRL. The SCLK output clock and the input sampling clock are both delayed versions of QSPInCLK. The delays are controlled by PHYCONFIGTXDLLDELAY and PHY-CONFIGRXDLLDELAY in QSPIn_PHYCONFIGURATION, but it is strongly recommended to use the provided settings from the data-sheet. This is the only way to guarantee functional operation and performance.

The sampling clock may optionally be a delayed version of the DQS input. This is configured by setting DQSENABLE bit in QSPIn_RDDATACAPTURE. The delay is configured by the same setting as used for QSPInCLK option. The use of DQS is discour-aged as functional operation and performance are not guaranteed.

The QSPI can operate in SDR (Single Data Rate) or DDR (Double Data Rate). For SDR, data is set up on negative edge of SCLK and sampled on positive edge of SCLK. For read instructions, the direction of data pins can switch immediately on a negative SCLK edge, or a number of dummy cycles can be added between writing data and reading data.

For DDR, data is set up and sampled on both positive and negative edges of SCLK. For read instructions, the QSPI always samples the first data on the positive edge of SCLK. Because transmitted data is always an integer number of bytes, there is typically an even number of output cycles before the read cycles start. This implies that SCLK is low when all output is transmitted, and ideally QSPI could start sampling data on the next positive edge of SCLK. However, the SPI slave needs to decode the last sample before being able to generate a new output on the first negative SCLK edge. Because of this, QSPI should be configured with one additional dummy cycle whenever DDR is used. For octal mode, however, one output cycle transimts an entire byte, making it possible that there is an odd number of output cycles. In this case, SCLK is high when all output is transmitted. This means that SPI slave can generate its first output on the next negative edge of SCLK allowing QSPI to sample at the first postivie edge after output is transmitted. In this case there is no need for any additional dummy cycles.

Some SPI slaves may start outputting data on positive SCLK edge rather than on negative SCLK edge. The only way to support this is by using DQS.

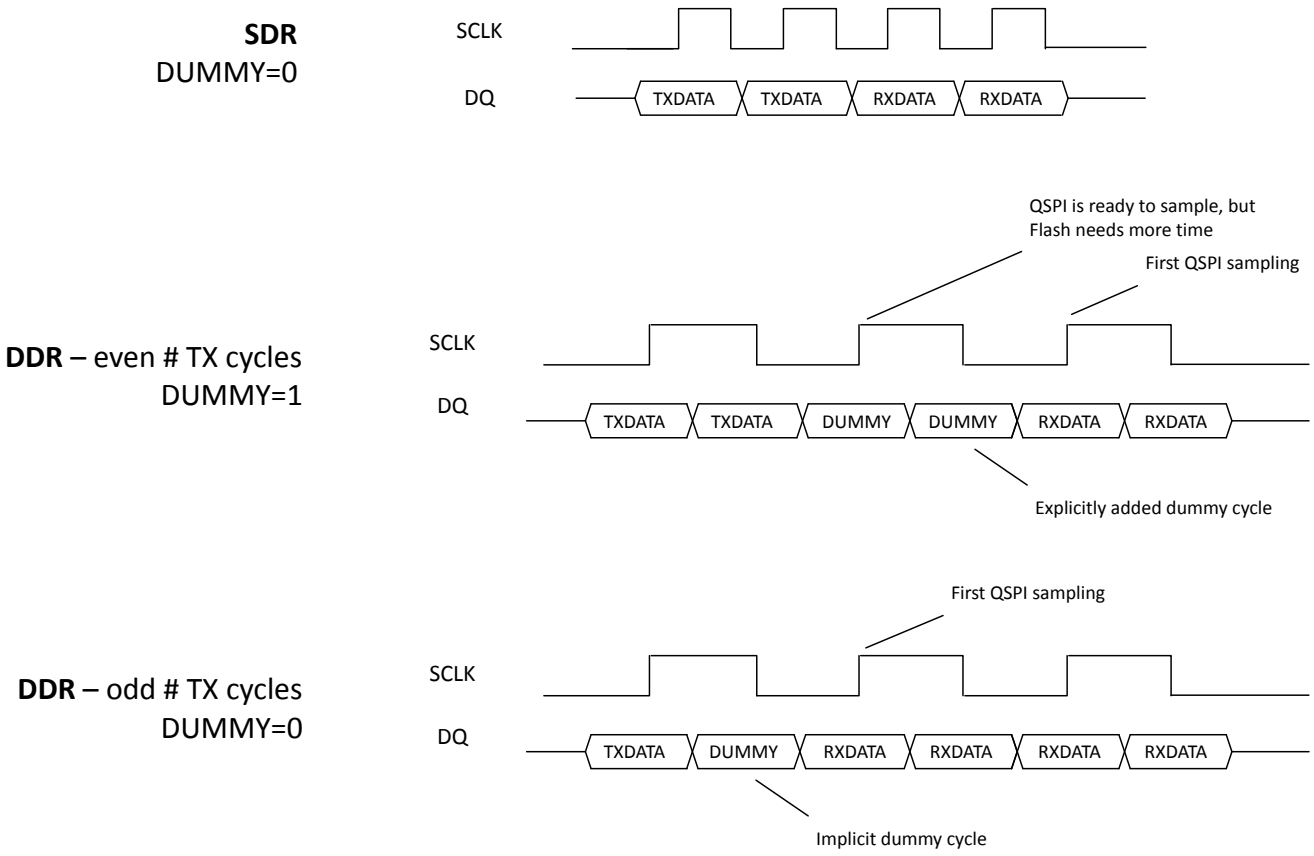


Figure 37.4. SDR and DDR I/O timing examples

37.3.10 Configure QSPI

In order to set up the QSPI according to the connected flash device, the following sequence can be followed:

1. Poll the IDLE bit in QSPIn_CONFIG to make sure the controller is idle.
2. Disable the QSPI controller by clearing the ENBSPI bit.
3. Update the QSPIn_DEVINSTRRDCONFIG and QSPIn_DEVINSTRWRCONFIG to configure the DAC/INDAC SPI transfers.
4. Optionally enable mode bits (if required) by setting MODEBITENABLE in QSPIn_DEVINSTRRDCONFIG and configure the QSPIn_MODEBITCONFIG register.
5. Update the QSPIn_DEVSIZECONFIG register to match the connected device.
6. Update the QSPIn_DEVDELAY register to tweak how the chip select is driven for each flash access.
7. Define the polling repetition delay in the POLLREPDELAY field in QSPIn_WRITECOMPLETIONCTRL.
8. Update the QSPIn_REMAPADDR. Affects DAC path only. Refer to 37.3.5.1 External Flash Address for further details.
9. Setup and enable write protection registers (QSPIn_LOWERWRPROT, QSPIn_UPPERWRPROT, QSPIn_WRPROTCTRL).
10. Enable required interrupts via QSPIn_IRQMASK.
11. Select I/O location in QSPIn_ROUTELOC0 and enable clock-output-pin, needed data pins and needed chip selects in QSPIn_ROUTEPEN.
12. Set PHYMODEENABLE in QSPIn_CONFIG and update PHYCONFIGTXDLLDELAY and PHYCONFIGRXDLLDELAY in QSPIn_PHYCONFIGURATION register based on SDR/DDR and I/O location. Remember to set PHYCONFIGRESYNC afterwards to load new DLL settings.
13. Select oscillator for QSPInCLK in CMU. Oscillator frequency controls SCLK frequency directly.
14. Optionally enable INDAC/DAC and enable QSPI Controller in the QSPIn_CONFIG register.

37.3.11 Code Execution

The QSPI allows the CPU to execute code directly from external flash memory. In order to make the code execution as efficient as possible, the code space memory region should be used (see 37.3.5 Memory Space), the controller should be configured for the highest throughput supported by the flash device and XIP should be enabled (if supported by the flash).

Note: Code execution is not supported in Octal mode unless non-PHY mode is used, see 37.3.15 PHY Octal Mode Limitations.

37.3.11.1 XIP Mode

The purpose of XIP mode is to reduce overhead during reads. When using XIP, no opcode is sent. The controller issues the address directly and the memory device responds with the data. XIP mode is supported in most flash devices. However, flash manufacturers do not have a consistent standard for implementing it. A common approach is to use signature bits that are sent to the device immediately following the address bytes.

37.3.11.2 Entering XIP Mode

How to enter XIP mode will vary according to the flash device. The general approach is described below. See the flash device data sheet for details.

- Make sure the flash device is in XIP mode
- Wait for the QSPI to be idle - poll the IDLE bit in QSPIn_CONFIG
- Program the MODE field in QSPIn_MODEBITCONFIG register according to the device
- Set the MODEBITENABLE bit in QSPIn_DEVINSTRRDCONFIG
- Set the ENTERXIPMODE bit in QSPIn_CONFIG

Note that when the MODEBITENABLE bit is set the QSPI will send 8 extra bits after the address and before the dummy cycles. This might require that the DUMMYRDCLKCYCLES field in QSPIn_DEVINSTRRDCONFIG is reduced to hit the correct read cycle for the flash. E.g. in QUAD mode, the DUMMYRDCLKCYCLES should typically be reduced by 2 when going from non-XIP to XIP mode.

37.3.11.3 Exiting XIP Mode

To exit XIP mode, do the following:

- Wait for the IDLE bit in QSPIn_CONFIG to go high
- Clear both ENTERXIPMODEIMM and ENTERXIPMODE in QSPIn_CONFIG
- Set INSTRTYPE in QSPIn_DEVINSTRRDCONFIG to 0
- Set the MODE field in QSPIn_MODEBITCONFIG to any value that will cause the flash to exit XIP mode. See the device data sheet.

- Do a dummy read. This will clear XIP mode both in the QSPI and the flash
- Wait for the IDLE bit in QSPIn_CONFIG to go high
- Re-configure the QSPI with correct settings for non-XIP mode

37.3.12 Servicing Interrupts

Refer to the QSPIn_IRQSTATUS register for details of all the interrupt sources provided. When an interrupt occurs the bit is set in QSPIn_IRQSTATUS. Interrupts are cleared by writing 1 to the same bit. An interrupt is enabled by setting the corresponding bit in QSPIn_IRQMASK.

37.3.13 Non-PHY mode

The QSPI controller is capable of running in a non-PHY mode where the SCLK output is divided down from the QSPInCLK. The clock division ratio must be at least 4 for SDR and 8 for DDR. This means that maximum performance is limited by maximum internal clock frequency and power is increased compared to normal PHY mode because clock frequency must be higher for the same SCLK frequency. Because of this, the use of non-PHY mode is discouraged, but is still documented here because it may be useful in some corner cases, e.g. to overcome the limitations in Octal mode.

When operating in non-PHY mode, the delay elements mentioned in 37.3.9 I/O Timing are not used. Default sampling is on positive edge of SCLK for SDR and in the middle to two SCLK edges for DDR, however, the sampling can be delayed by a configurable number of QSPInCLK cycles. By default, output data is generated on negative edge of SCLK for SDR and one QSPInCLK cycle after an SCLK edge for DDR. For DDR, the output can be delayed by a configurable number of QSPInCLK cycles. The delays are configured by DELAY and DDRREADDELAY fields in QSPIn_RDDATACAPTURE register. SPI Mode 0 is the deault mode, but for SDR, other modes can be selected when using non-PHY mode. This is done by configuring SELCLKPHASE and SELCLKPOL in QSPIn_CTRL regsiter.

37.3.14 Frequency Limitations

The QSPI has two input clocks: HFBUSCLK and QSPInCLK. The HFBUSCLK is used by the system to read and write data and configuration to the QSPI. The QSPInCLK is used to generate the serial clock (SCLK) which goes to the external flash. The HFBUSCLK frequency depends on the HFCLK frequency and the CMU_HFBUSPRESC register. The maximum allowed frequency for HFBUSCLK is 50 MHz.

$$f_{HFBUSCLK} = f_{HFCLK} / (CMU\_HFBUSPRESC.PRESC + 1)$$

Figure 37.5. HFBUSCLK Frequency

The QSPInCLK source is controlled by QSPInCLKSEL in CMU_QSPICTRL, see 10.3.1.7 QSPInCLK - QSPI Reference Clock. The maximum allowed frequency for QSPInCLK is 50 MHz.

There are some restrictions on frequency relationships between HFBUSCLK and QSPInCLK. Failure to obey these restrictions may lead to currupt data.

When $f_{HFBUSCLK} > f_{QSPInCLK}$:

$$f_{HFBUSCLK} < f_{QSPInCLK} * spi\_clk\_divider * ahb\_data\_size / spi\_data\_size$$

When $f_{HFBUSCLK} < f_{QSPInCLK}$:

$$f_{HFBUSCLK} > f_{QSPInCLK} * spi\_data\_size / (ahb\_data\_size * spi\_clk\_divider)$$

When PHY mode is enabled:

$$f_{QSPInCLK} / 3.33 < f_{HFBUSCLK} < f_{QSPInCLK} * 10$$

When non-PHY mode is enabled:

$$f_{QSPInCLK} / 25 < f_{HFBUSCLK} < f_{QSPInCLK} * 25$$

Where:

ahb_data_size = Number of bytes in AHB transfers

spi_data_size = Number of bytes per SPI cycle. E.g. QUAD DDR will have 2 times 4 bit = 1 byte.

spi_clk_divider = Clock division configured by MSTRBAUDDIV in QSPIn_CONFIG register. For PHY-mode, spi_clk_divider is always 1.

Figure 37.6. Frequency Limits

37.3.15 PHY Octal Mode Limitations

It is possible to use the QSPI in Octal mode with the PHY mode enabled. However, this mode imposes some extra limitations. When using PHY and Octal mode, read/write operations must be done in "bursts". A burst is defined here as

- Each address must be word (4-byte) aligned
- Each address must be equal to the previous address + 4 (next word-aligned address)
- Each access must be of the same type as the previous (read/write)

Between each burst, software must wait for the QSPI to be idle by polling the IDLE bit in QSPIn_CONFIG before starting another burst.

Note: This implies that code execution from external flash is *not* supported in this mode

37.3.16 SPI Legacy Mode

SPI legacy mode allows software to access the internal TX-FIFO and RX-FIFO directly, thus bypassing the DAC, INDAC and STIG. Legacy mode is enabled by setting the ENBLEGACYIPMODE bit in QSPIn_CONFIG. In legacy mode, a write to any address in the QSPI memory region will push the TX-FIFO. Any read will pop the RX-FIFO.

The TX and RX FIFOs are both 5 bytes deep. Software must make sure that the TX and RX FIFOs don't overflow or underflow. Interrupts are provided to indicate when the fill levels pass programmable watermarks, which are themselves programmable in the registers QSPIn_TXTHRESH and QSPIn_RXTHRESH.

37.3.17 QSPI Pin Configurations

Table 37.1 QSPI Pins on page 1400 shows different routing options for QSPI pins and also states GPIO mode configuration requirements for each QSPI pins (Please note that GPIO mode configuration is required only when corresponding QSPI pin is enabled via QSPIn_ROUTEPEN).

Table 37.1. QSPI Pins

QSPI Pins	ROUTELOC0	ROUTELOC1	GPIO Mode
SCLK	F6[1]	E14	PUSHPULL / PUSHPULLALT
DQ0	D9[1]	A2	PUSHPULL / PUSHPULLALT
DQ1	D10[1]	A3	PUSHPULL / PUSHPULLALT
DQ2	D11[1]	A4	PUSHPULL / PUSHPULLALT
DQ3	D12[1]	A5	PUSHPULL / PUSHPULLALT
DQ4	E8[1]	B3	PUSHPULL / PUSHPULLALT
DQ5	E9[1]	B4	PUSHPULL / PUSHPULLALT
DQ6	E10[1]	B5	PUSHPULL / PUSHPULLALT
DQ7	E11[1]	B6	PUSHPULL / PUSHPULLALT
CS0	F7[1]	A0	PUSHPULL / PUSHPULLALT
CS1	F8[1]	A1	PUSHPULL / PUSHPULLALT
DQS	F9[1]	E15	INPUT
RST0	E14[1]	C2	PUSHPULL / PUSHPULLALT
RST1	E15[1]	C3	PUSHPULL / PUSHPULLALT

Note:
1. High speed location

37.4 Register Map

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

Offset	Name	Type	Description
0x000	QSPIn_CONFIG	RWH	Octal-SPI Configuration Register
0x004	QSPIn_DEVINSTRRDCONFIG	RW	Device Read Instruction Configuration Register
0x008	QSPIn_DEVINSTRWRCONFIG	RW	Device Write Instruction Configuration Register
0x00C	QSPIn_DEVDELAY	RW	Device Delay Register
0x010	QSPIn_RDDATACAPTURE	RW	Read Data Capture Register
0x014	QSPIn_DEVSIZECONFIG	RW	Device Size Configuration Register
0x018	QSPIn_SRAMPARTITIONCFG	RW	SRAM Partition Configuration Register
0x01C	QSPIn_INDAHBADDRTRIGGER	RW	Indirect Address Trigger Register
0x024	QSPIn_REMAPADDR	RW	Remap Address Register
0x028	QSPIn_MODEBITCONFIG	RWH	Mode Bit Configuration Register
0x02C	QSPIn_SRAMFILL	R	SRAM Fill Register
0x030	QSPIn_TXTHRESH	RW	TX Threshold Register
0x034	QSPIn_RXTHRESH	RW	RX Threshold Register
0x038	QSPIn_WRITECOMPLE-TIONCTRL	RW	Write Completion Control Register
0x03C	QSPIn_NOOFPOLLSBEFEXP	RW	Polling Expiration Register
0x040	QSPIn_IRQSTATUS	RWH	Interrupt Status Register
0x044	QSPIn_IRQMASK	RW	Interrupt Mask
0x050	QSPIn_LOWERWRPROT	RW	Lower Write Protection Register
0x054	QSPIn_UPPERWRPROT	RW	Upper Write Protection Register
0x058	QSPIn_WRPROTCTRL	RW	Write Protection Control Register
0x060	QSPIn_INDIRECTREADX-FERCTRL	RWH	Indirect Read Transfer Control Register
0x064	QSPIn_INDIRECTREADXFER-WATERMARK	RW	Indirect Read Transfer Watermark Register
0x068	QSPIn_INDIRECTREADXFER-START	RW	Indirect Read Transfer Start Address Register
0x06C	QSPIn_INDIRECTREADXFER-NUMBYTES	RW	Indirect Read Transfer Number Bytes Register
0x070	QSPIn_INDIRECTWRITEX-FERCTRL	RWH	Indirect Write Transfer Control Register
0x074	QSPIn_INDIRECTWRITEXFER-WATERMARK	RW	Indirect Write Transfer Watermark Register
0x078	QSPIn_INDIRECTWRITEXFER-START	RW	Indirect Write Transfer Start Address Register
0x07C	QSPIn_INDIRECTWRITEXFER-NUMBYTES	RW	Indirect Write Transfer Number Bytes Register
0x080	QSPIn_INDIRECTTRIGGER-ADDRRANGE	RW	Indirect Trigger Address Range Register

Offset	Name	Type	Description
0x08C	QSPIn_FLASHCOMMANDCTRL-MEM	RWH	Flash Command Control Memory Register (STIG)
0x090	QSPIn_FLASHCMDCTRL	RWH	Flash Command Control Register (STIG)
0x094	QSPIn_FLASHCMDADDR	RW	Flash Command Address Register (STIG)
0x0A0	QSPIn_FLASHRDDATALOWER	R	Flash Command Read Data Register (Lower) (STIG)
0x0A4	QSPIn_FLASHRDDATAUPPER	R	Flash Command Read Data Register (Upper) (STIG)
0x0A8	QSPIn_FLASHWRDATALOWER	RW	Flash Command Write Data Register (Lower) (STIG)
0x0AC	QSPIn_FLASHWRDATAUPPER	RW	Flash Command Write Data Register (Upper) (STIG)
0x0B0	QSPIn_POLLINGFLASHSTATUS	RWH	Polling Flash Status Register
0x0B4	QSPIn_PHYCONFIGURATION	RW	PHY Configuration Register
0x0E0	QSPIn_OPCODEEXTLOWER	RW	Opcode Extension Register (Lower)
0x0E4	QSPIn_OPCODEEXTUPPER	RW	Opcode Extension Register (Upper)
0x0FC	QSPIn_MODULEID	R	Module ID Register
0x104	QSPIn_ROUTEPEN	RW	I/O Routing Pin Enable Register
0x108	QSPIn_ROUTELOC0	RW	I/O Route Location Register 0

37.5 Register Description

37.5.1 QSPIn_CONFIG - Octal-SPI Configuration 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	1	0	0				0	0	0	0xF				0	0	0		0			0x0		0	0	1	0	0	0	0	0	1
Access	R	RW	RW				RW	RW	RW	RW				RW	RW	RW		RW			RW		RW	RW	RW	RW	RW	RW	RW	RW	RW
Name	IDLE	DUALBYTEOPCODEEN	CRCENABLE				PIPELINEPHY	ENABLEDTRPROTOCOL	ENABLEAHBDECODER	MSTRBAUDDIV				ENTERXIPMODEIMM	ENTERXIPMODE	ENBAHBADDRREMAP		WRPROTFLASH			PERIPHCSLINES		PERIPHSELDEC	ENBLEGACYIPMODE	ENBDIRACCCTLR	DEVRSTCONFIG	ENBDEVRST	ENBDEVHOLD	PHYMODEENABLE	SELCLKPHASE	SELCLKPOL	ENBSPI

Bit	Name	Reset	Access	Description
31	IDLE	1	R	**Serial Interface and Low Level SPI Pipeline is IDLE**
	This is a STATUS read-only bit.			
30	DUALBYTEOP-CODEEN	0	RW	**Dual-byte Opcode Mode Enable Bit**
	This bit is to be set in case the target Flash Device supports dual byte opcode. It is applicable for octal I/O mode only. If enabled, the supplementing bytes are taken from QSPIn_OPCODEEXTLOWER and QSPIn_OPCODEEXTUPPER			
29	CRCENABLE	0	RW	**CRC Enable Bit**
	This bit is to be set in case the target flash device supports CRC. Applicable for octal DDR protocol only.			
28:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	PIPELINEPHY	0	RW	**Pipeline PHY Mode Enable**
	PHY pipeline mode is not supported. This bit should never be set.			
24	ENABLEDTRPRO-TOCOL	0	RW	**Enable DTR Protocol**
	When this bit is set, all transfers use DDR. If the bit is not set, address and data phase of data read instructions will still use DDR if DDREN field of QSPIn_DEVINSTRRDCONFIG register is set.			
23	ENABLEAHBDE-CODER	0	RW	**Enable Address Decoder**
	When set to 0 active slave is selected based on PERIPHCSLINES and PERIPHSELDEC. When set to 1 active slave is selected based on address (the partition for each device is calculated based on QSPIn_DEVSIZECONFIG) and PERIPH-CSLINES and PERIPHSELDEC are not used. Address Decoder is only supported for Direct Access Controller.			
22:19	MSTRBAUDDIV	0xF	RW	**Master Mode Baud Rate Divisor**
	This field is only valid for non-PHY mode. The SPI baud rate (SCLK frequency) is equal to QSPInCLK / 2*(MSTRBAUDDIV +1). The resulting divisor, 2*(MSTRBAUDDIV+1), must be at least 4 for SDR and at least 8 for DDR transfers.			

Bit	Name	Reset	Access	Description
18	ENTERXIPMO-DEIMM	0	RW	**Enter XIP Mode Immediately**
	If XIP is enabled, then setting to 0 will cause the controller to exit XIP mode on the next READ instruction. When set to 1, operate the device in XIP mode immediately. Use this register when the external device wakes up in XIP mode. The controller will assume the next READ instruction will be passed to the device as an XIP instruction, and therefore will not require the READ opcode to be transferred.			
17	ENTERXIPMODE	0	RW	**Enter XIP Mode on Next READ**
	If XIP is enabled, then setting to 0 will cause the controller to exit XIP mode on the next READ instruction. If XIP is disabled, then setting to 1 will inform the controller that the device is ready to enter XIP on the next READ instruction.			
16	ENBAHBADDRRE-MAP	0	RW	**Enable Address Remapping**
	When set to 1, the incoming internal address will be translated to (address + N) before sending to flash, where N is the value stored in the remap address register. Address Remapping is only supported for Direct Access Controller.			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	WRPROTFLASH	0	RW	**Write Protect Flash Pin**
	Set to drive the Write Protect pin of the FLASH device. Note that the WP pin is only valid in SINGLE or DUAL transfer modes.			
13:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:10	PERIPHCSLINES	0x0	RW	**Peripheral Chip Select Lines**
	If PERIPHSELDEC = 0 write 0 to select CS0 or 1 to select CS1. If PERIPHSELDEC = 1 the value of this register is output directly on [CS1,CS0] (allows external 2-to-4 decoder)			
9	PERIPHSELDEC	0	RW	**Peripheral Select Decode**
	When 0 only one chip select is active. When 1 the chip selects (CS1, CS0) are driven from PERIPHCSLINES (allows external 2-to-4 decoder)			
8	ENBLEGACYIP-MODE	0	RW	**Legacy IP Mode Enable**
	0 : Use DAC/INDAC/STIG. 1 : Use legacy SPI mode.			
7	ENBDIRACCCTLR	1	RW	**Enable Direct Access Controller**
	0 : disable DAC once current transfer is complete. 1 : enable DAC. When both DAC and INDAC are disabled, memory accesses result in a bus fault.			
6	DEVRSTCONFIG	0	RW	**Device Reset Configuration**
	RESET pin configuration. 0 : RESET feature on DQ3 pin of the Flash device. 1 : RESET feature on dedicated pin of the Flash device			
5	ENBDEVRST	0	RW	**Enable Device Reset**
	Set to drive the RESET pin of the FLASH device and reset for de-activation of the RESET pin feature			
4	ENBDEVHOLD	0	RW	**Enable Device Hold**
	Set to drive the HOLD pin of the FLASH device and reset for deactivation of the HOLD pin feature			
3	PHYMODEENABLE	0	RW	**PHY Mode Enable**
	PHY mode enable			
2	SELCLKPHASE	0	RW	**Clock Phase, CPHA**
	Selects which edge to capture data on. Must be zero for DDR modes and PHY mode.			

Bit	Name	Reset	Access	Description
1	SELCLKPOL	0	RW	**Clock Polarity, CPOL**
	0 : the SPI clock is idle low. 1 : the SPI clock is idle high. Must be zero for DDR modes and PHY mode.			
0	ENBSPI	1	RW	**QSPI Enable**
	Write 0 to disable the QSPI once current transfer of the data word is complete. Write 1 to enable the QSPI			

37.5.2 QSPIn_DEVINSTRRDCONFIG - Device Read Instruction 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				0x00								0			0x0				0x0			0	0x0		0x03						
Access				RW								RW			RW				RW			RW	RW		RW						
Name				DUMMYRDCLKCYCLES								MODEBITENABLE			DATAXFERTYPEEXTMODE				ADDRXFERTYPESTDMODE			DDREN	INSTRTYPE		RDOPCODENONXIP						

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	DUMMYRDCLKCYCLES	0x00	RW	**Dummy Read Clock Cycles** Dummy Read Clock Cycles: Number of dummy clock cycles required by device for read instruction.
23:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
20	MODEBITENABLE	0	RW	**Mode Bit Enable** Set this field to 1 to ensure that the mode bits as defined in the QSPIn_MODEBITCONFIG register are sent following the address bytes for normal reads.
19:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	DATAXFERTYPEEXTMODE	0x0	RW	**Data Transfer Type for Standard SPI Modes** 0: Single mode. 1: Dual mode. 2: Quad mode. 3: Octal mode. This field is only valid when INSTRTYPE = 0.
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	ADDRXFERTYPESTDMODE	0x0	RW	**Address Transfer Type for Standard SPI Modes** 0: Single mode. 1: Dual mode. 2: Quad mode. 3: Octal mode. This field is only valid when INSTRTYPE = 0.
11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10	DDREN	0	RW	**DDR Enable** This field should be set (and ENABLEDTRPROTOCOL in QSPIn_CONFIG should be cleared) when opcode from RDOPCODENONXIP supports DDR for address and data phase only.
9:8	INSTRTYPE	0x0	RW	**Instruction Type** 0: Single mode. 1: Dual mode. 2: Quad mode. 3: Octal mode. If this is non-zero, address and data modes are also taken from this field. This field is also used for write operations.

Bit	Name	Reset	Access	Description
7:0	RDOPCODENONXIP	0x03	RW	**Read Opcode in Non-XIP Mode**
	Read Opcode to use when not in XIP mode			

37.5.3 QSPIn_DEVINSTRWRCONFIG - Device Write Instruction 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				0x00											0x0				0x0					0	0x02							
Access				RW											RW				RW					RW	RW							
Name				DUMMYWRCLKCYCLES											DATAXFERTYPEEXTMODE				ADDRXFERTYPESTDMODE					WELDIS	WROPCODE							

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	DUMMYWRCLKCYCLES	0x00	RW	**Dummy Write Clock Cycles**
	Dummy Write Clock Cycles: Number of dummy clock cycles required by device for write instruction.			
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	DATAXFERTYPEEXTMODE	0x0	RW	**Data Transfer Type for Standard SPI Modes**
	0: Single mode. 1: Dual mode. 2: Quad mode. 3: Octal mode. This field is only valid when INSTRTYPE = 0.			
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:12	ADDRXFERTYPESTDMODE	0x0	RW	**Address Transfer Type for Standard SPI Modes**
	0: Single mode. 1: Dual mode. 2: Quad mode. 3: Octal mode. This field is only valid when INSTRTYPE = 0.			
11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	WELDIS	0	RW	**WEL Disable**
	This is to turn off automatic issuing of WEL Command before write operation for DAC or INDAC			
7:0	WROPCODE	0x02	RW	**Write Opcode**
	Write Opcode			

37.5.4 QSPIn_DEVDELAY - Device Delay Register

This register is used to introduce relative delays into the generation of the master output signals. All timings are defined in cycles of QSPInCLK.

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	0x00								0x00								0x00								0x00							
Access	RW								RW								RW								RW							
Name	DNSS								DBTWN								DAFTER								DINIT							

Bit	Name	Reset	Access	Description
31:24	DNSS	0x00	RW	**Clock Delay for Chip Select Deassert**
	Delay in QSPInCLK cycles for the length that the master mode chip select outputs are de-asserted between transactions. The minimum delay is always SCLK period to ensure the chip select is never re-asserted within an SCLK period.			
23:16	DBTWN	0x00	RW	**Clock Delay Between Two Chip Selects**
	Delay in QSPInCLK cycles between one chip select being de-activated and the activation of another. This is used to ensure a quiet period between the selection of two different slaves and requires the transmit FIFO to be empty.			
15:8	DAFTER	0x00	RW	**Clock Delay for Last Transaction Bit**
	Delay in QSPInCLK cycles between last bit of current transaction and deasserting CSx. By default, the chip select will be deasserted on the cycle following the completion of the current transaction.			
7:0	DINIT	0x00	RW	**Clock Delay for CS**
	Delay in QSPInCLK cycles between setting CSx low and first bit transfer.			

37.5.5 QSPIn_RDDATACAPTURE - Read Data Capture 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											0				0x0				1
Access													RW											RW				RW				RW
Name													DDRREADDELAY											DQSENABLE				DELAY				BYPASS

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	DDRREADDELAY	0x0	RW	**DDR Read Delay** This field is only valid for non-PHY DDR mode. Delay the transmitted data by the programmed number of QSPInCLK cycles.
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	DQSENABLE	0	RW	**DQS Enable Bit** If enabled, a delayed version of DQS input is used as sampling clock. This mode is not recommended and it is not available for non-PHY mode.
7:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4:1	DELAY	0x0	RW	**Read Delay** Delay the read data capturing logic by the programmed number of QSPInCLK cycles. This field should be set to 0 when PHY mode is enabled.
0	BYPASS	1	RW	**Bypass the Adapted Loopback Clock Circuit** This bit should always be set.

37.5.6 QSPIn_DEVSIZECONFIG - Device Size Configuration 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				0x10									0x100								0x2	
Access								RW		RW				RW									RW								RW	
Name								MEMSIZEONCS1		MEMSIZEONCS0				BYTESPERSUBSECTOR									BYTESPERDEVICEPAGE								NUMADDRBYTES	

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*		
24:23	MEMSIZEONCS1	0x0	RW	**Size of Flash Device Connected to CS[1] Pin** Value=00 : size of 512Mb(64MB). Value=01 : size of 1Gb(128MB). Value=10 : size of 2Gb(256MB). Value=11 : size of 4Gb(512MB).
22:21	MEMSIZEONCS0	0x0	RW	**Size of Flash Device Connected to CS[0] Pin** Value=00 : size of 512Mb(64MB). Value=01 : size of 1Gb(128MB). Value=10 : size of 2Gb(256MB). Value=11 : size of 4Gb(512MB).
20:16	BYTESPERSUB-SECTOR	0x10	RW	**Number of Bytes Per Block** Number of bytes per Block. This is required by the controller for performing the write protection logic. The number of bytes per block must be a power of 2 number.
15:4	BYTESPERDEVICE-PAGE	0x100	RW	**Number of Bytes Per Device Page** Number of bytes per device page. This is required by the controller for performing FLASH writes up to and across page boundaries.
3:0	NUMADDRBYTES	0x2	RW	**Number of Address Bytes** Number of address bytes. A value of 0 indicates 1 byte.

37.5.7 QSPIn_SRAMPARTITIONCFG - SRAM Partition Configuration 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																									0x80							
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*		
7:0	ADDR	0x80	RW	**Indirect Read Partition Size**
	Defines the size of the indirect read partition in the SRAM, in units of SRAM locations. By default, half of the SRAM is reserved for indirect read operation, and half for indirect write.			

37.5.8 QSPIn_INDAHBADDRTRIGGER - Indirect Address Trigger 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																0x00000000																
Access																RW																
Name																ADDR																

Bit	Name	Reset	Access	Description
31:0	ADDR	0x00000000	RW	**Indirect Address Trigger Register**
	This is the base address that will be used by the AHB controller. When the incoming read access address matches a range of addresses from this trigger address to the trigger address + the range defined by QSPIn_INDIRECTTRIGGERADDRRANGE, then the AHB request will be completed by fetching data from the Indirect Controllers SRAM.			

37.5.9 QSPIn_REMAPADDR - Remap Address 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	0x00000000																															
Access	RW																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	RW	**Remap Address Value**
	This register is used to remap an internal address to a different address used by the FLASH device.			

37.5.10 QSPIn_MODEBITCONFIG - Mode Bit Configuration 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	0x00								0x00								0						0x2		0x00							
Access	R								R								RW						RW		RW							
Name	RXCRCDATALOW								RXCRCDATAUP								CRCOUTENABLE						CHUNKSIZE		MODE							

Bit	Name	Reset	Access	Description
31:24	RXCRCDATALOW	0x00	R	**RX CRC Data (lower)**
	The first CRC byte returned after RX data chunk.			
23:16	RXCRCDATAUP	0x00	R	**RX CRC Data (upper)**
	The second CRC byte returned after RX data chunk.			
15	CRCOUTENABLE	0	RW	**CRC# Output Enable Bit**
	When enabled, the controller expects the Flash Device to toggle CRC data on both SPI clock edges in CRC->CRC# sequence and calculates CRC compliance accordingly.			
14:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	CHUNKSIZE	0x2	RW	**Chunk Size**
	Defines size of chunk after which CRC data is expected to show up on the SPI interface for write and read data transfers.			
7:0	MODE	0x00	RW	**Mode Bits**
	These are the 8 mode bits that are sent to the device following the address bytes if mode bit transmission has been enabled.			

37.5.11 QSPIn_SRAMFILL - SRAM Fill 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	0x0000																0x0000															
Access	R																R															
Name	SRAMFILLINDACWRITE																SRAMFILLINDACREAD															

Bit	Name	Reset	Access	Description
31:16	SRAMFILLINDAC-WRITE	0x0000	R	**SRAM Fill Level (Indirect Write Partition)**
	Identifies the current fill level of the SRAM Indirect Write partition. Unit is words (4 bytes).			
15:0	SRAMFILLINDAC-READ	0x0000	R	**SRAM Fill Level (Indirect Read Partition)**
	Identifies the current fill level of the SRAM Indirect Read partition. Unit is words (4 bytes).			

37.5.12 QSPIn_TXTHRESH - TX Threshold 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																														0x01		
Access																														RW		
Name																														LEVEL		

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	LEVEL	0x01	RW	**Threshold Level**
	This field is only to be used for legacy SPI mode. Defines the level at which the small TX FIFO not full interrupt is generated			

37.5.13 QSPIn_RXTHRESH - RX Threshold 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																												0x01				
Access																												RW				
Name																												LEVEL				

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	LEVEL	0x01	RW	**Threshold Level** This field is only to be used for legacy SPI mode. Defines the level at which the small RX FIFO not empty interrupt is generated

37.5.14 QSPIn_WRITECOMPLETIONCTRL - Write Completion Control Register

This register defines how the controller will poll the device following a write transfer

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	0x00								0x01								0	0	0			0x0			0x05							
Access	RW								RW								RW	RW	RW			RW			RW							
Name	POLLREPDELAY								POLLCOUNT								ENABLEPOLLINGEXP	DISABLEPOLLING	POLLINGPOLARITY			POLLINGBITINDEX			OPCODE							

Bit	Name	Reset	Access	Description
31:24	POLLREPDELAY	0x00	RW	**Poll Repetition Delay**
	Defines additional delay for maintain Chip Select de-asserted during auto-polling phase			
23:16	POLLCOUNT	0x01	RW	**Poll Count**
	Defines the number of times the controller should expect to see a true result from the polling in successive reads of the device register.			
15	ENABLEPOLLIN-GEXP	0	RW	**Enable Polling Expiration**
	Set to '1' to enable auto-polling expiration.			
14	DISABLEPOLLING	0	RW	**Disable Polling**
	This switches off the automatic polling function			
13	POLLINGPOLARITY	0	RW	**Polling Polarity**
	Defines the polling polarity. If '1', then the write transfer to the device will be complete if the polled bit is equal to '1'. If '0', then the write transfer to the device will be complete if the polled bit is equal to '0'.			
12:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	POLLINGBITINDEX	0x0	RW	**Polling Bit Index**
	Defines the bit index that should be polled. A value of 010 means that bit 2 of the returned data will be polled for.A value of 111 means that bit 7 of the returned data will be polled for.			
7:0	OPCODE	0x05	RW	**Opcode**
	Defines the opcode that should be issued by the controller when it is automatically polling for device program completion. This command is issued followed all device write operations. By default, this will poll the standard device STATUS register using opcode 0x05			

37.5.15 QSPIn_NOOFPOLLSBEFEXP - Polling Expiration 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	0xFFFFFFFF																															
Access	RW																															
Name	NOOFPOLLSBEFEXP																															

Bit	Name	Reset	Access	Description
31:0	NOOFPOLLSBE-FEXP	0xFFFFFFFF	RW	**Number of Polls Cycles Before Expiration**
	Number of polls cycles before expiration			

37.5.16 QSPIn_IRQSTATUS - Interrupt Status Register

The status fields in this register are set when the described event occurs and the interrupt is enabled in the mask register. When any of these bit fields are set, the interrupt output is asserted high. The fields are each cleared by writing a 1 to the field. Note that bit fields 7 through 11 are only valid when legacy SPI mode is active.

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														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access														RWH	RWH	RWH		RWH	RWH	RWH	RWH	RWH	RWH	RWH	RWH	RWH	RWH	RWH	RWH	RWH	RWH	RWH
Name														TXCRCCHUNKBRK	RXCRCDATAVAL	RXCRCDATAERR		STIGREQINT	POLLEXPINT	INDRDSRAMFULL	RXFIFOFULL	RXFIFONOTEMPTY	TXFIFOFULL	TXFIFONOTFULL	RECVOVERFLOW	INDIRECTXFERLEVELBREACH	ILLEGALACCESSDET	PROTWRATTEMPT	INDIRECTREADREJECT	INDIRECTOPDONE	UNDERFLOWDET	MODEMFAIL

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	TXCRCCHUNKBRK	0	RWH	**TX CRC Chunk Was Broken** This interrupt informs the system that program page SPI transfer was discontinued somewhere inside the chunk.
17	RXCRCDATAVAL	0	RWH	**RX CRC Data Valid** New RX CRC data was captured from Flash Device
16	RXCRCDATAERR	0	RWH	**RX CRC Data Error** RX CRC data error CRC data from Flash Device does not correspond to the one dynamically calculated by the controller.
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	STIGREQINT	0	RWH	**The Controller is Ready for Getting Another STIG Request** The controller is ready for getting another STIG request.
13	POLLEXPINT	0	RWH	**The Maximum Number of Programmed Polls Cycles is Expired** The maximum number of programmed polls cycles is expired
12	INDRDSRAMFULL	0	RWH	**Indirect Read Partition Overflow** Indirect Read Partition of SRAM is full and unable to immediately complete indirect operation
11	RXFIFOFULL	0	RWH	**Small RX FIFO Full** Current FIFO status can be ignored in non-SPI legacy mode. 0 : FIFO is not full. 1 : FIFO is full.
10	RXFIFONOTEMPTY	0	RWH	**Small RX FIFO Not Empty** Current FIFO status can be ignored in non-SPI legacy mode. 0 : FIFO has less than RX THRESHOLD entries, 1 : FIFO has >= THRESHOLD entries.
9	TXFIFOFULL	0	RWH	**Small TX FIFO Full** Current FIFO status can be ignored in non-SPI legacy mode 0 : FIFO is not full, 1 : FIFO is full

Bit	Name	Reset	Access	Description
8	TXFIFONOTFULL	0	RWH	**Small TX FIFO Not Full**
	Current FIFO status can be ignored in non-SPI legacy mode. 0 : FIFO has >= THRESHOLD entries, 1 : FIFO has less than THRESHOLD entries			
7	RECVOVERFLOW	0	RWH	**Receive Overflow**
	This should only occur in Legacy SPI mode. Set if an attempt is made to push the RX FIFO when it is full. This bit is reset only by a system reset and cleared only when this register is read. If a new push to the RX FIFO occurs coincident with a register read this flag will remain set. 0 : no overflow has been detected. 1 : an overflow has occurred.			
6	INDIRECTXFERLE-VELBREACH	0	RWH	**Indirect Transfer Watermark Level Breached**
	Indirect Transfer Watermark Level Breached			
5	ILLEGALACCESS-DET	0	RWH	**Illegal Memory Access Has Been Detected**
	Illegal memory access has been detected. AHB wrapping bursts and the use of SPLIT/RETRY accesses will cause this error interrupt to trigger.			
4	PROTWRATTEMPT	0	RWH	**Write to Protected Area Was Attempted and Rejected**
	Write to protected area was attempted and rejected.			
3	INDIRECTREADRE-JECT	0	RWH	**Indirect Operation Was Requested but Could Not Be Accepted**
	Indirect operation was requested but could not be accepted. Two indirect operations already in storage.			
2	INDIRECTOPDONE	0	RWH	**Indirect Operation Complete**
	Controller has completed last triggered indirect operation. This means that last data has been read from SRAM by software (read operation) or by QSPI controller (write operation).			
1	UNDERFLOWDET	0	RWH	**Underflow Detected**
	0 : no underflow has been detected. 1 : underflow is detected and an attempt to transfer data is made when the small TX FIFO is empty. This may occur when write data is being supplied too slowly to keep up with the requested write operation. This bit is reset only by a system reset and cleared only when the register is read.			
0	MODEMFAIL	0	RWH	**Mode M Failure**
	This interrupt is not in use.			

37.5.17 QSPIn_IRQMASK - Interrupt Mask

0 : the interrupt for the corresponding interrupt status register bit is disabled. 1 : the interrupt for the corresponding interrupt status register bit is enabled.

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														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Access														RW	RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name														TXCRCCHUNKBRKMASK	RXCRCDATAVALMASK	RXCRCDATAERRMASK		STIGREQMASK	POLLEXPINTMASK	INDRDSRAMFULLMASK	RXFIFOFULLMASK	RXFIFONOTEMPTYMASK	TXFIFOFULLMASK	TXFIFONOTFULLMASK	RECVOVERFLOWMASK	INDIRECTXFERLEVELBREACHMASK	ILLEGALACCESSDETMASK	PROTWRATTEMPTMASK	INDIRECTREADREJECTMASK	INDIRECTOPDONEMASK	UNDERFLOWDETMASK	MODEMFAILMASK

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	TXCRCCHUNKBRKMASK	0	RW	**TX CRC Chunk Was Broken Mask**
	TX CRC chunk was broken Mask			
17	RXCRCDATAVALMASK	0	RW	**RX CRC Data Valid Mask**
	RX CRC data valid Mask			
16	RXCRCDATAERRMASK	0	RW	**RX CRC Data Error Mask**
	RX CRC data error Mask			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	STIGREQMASK	0	RW	**STIG Request Completion Mask**
	STIG request completion Mask			
13	POLLEXPINTMASK	0	RW	**Polling Expiration Detected Mask**
	Polling expiration detected Mask			
12	INDRDSRAMFULLMASK	0	RW	**Indirect Read Partition Overflow Mask**
	Indirect Read Partition overflow mask			
11	RXFIFOFULLMASK	0	RW	**Small RX FIFO Full Mask**
	Small RX FIFO full Mask			

Bit	Name	Reset	Access	Description
10	RXFIFONOTEMPTY-MASK	0	RW	**Small RX FIFO Not Empty Mask**
	Small RX FIFO not empty Mask			
9	TXFIFOFULLMASK	0	RW	**Small TX FIFO Full Mask**
	Small TX FIFO full Mask			
8	TXFIFONOTFULL-MASK	0	RW	**Small TX FIFO Not Full Mask**
	Small TX FIFO not full Mask			
7	RECVOVERFLOW-MASK	0	RW	**Receive Overflow Mask**
	Receive Overflow Mask			
6	INDIRECTXFERLE-VELBREACHMASK	0	RW	**Transfer Watermark Breach Mask**
	Transfer Watermark Breach Mask			
5	ILLEGALACCESS-DETMASK	0	RW	**Illegal Access Detected Mask**
	Illegal Access Detected Mask			
4	PROTWRATTEMPT-MASK	0	RW	**Protected Area Write Attempt Mask**
	Protected Area Write Attempt Mask			
3	INDIRECTREADRE-JECTMASK	0	RW	**Indirect Read Reject Mask**
	Indirect Read Reject Mask			
2	INDIRECTOPDONE-MASK	0	RW	**Indirect Complete Mask**
	Indirect Complete Mask			
1	UNDERFLOWDET-MASK	0	RW	**Underflow Detected Mask**
	Underflow Detected Mask			
0	MODEMFAILMASK	0	RW	**Mode M Failure Mask**
	Mode M Failure Mask			

37.5.18 QSPIn_LOWERWRPROT - Lower Write Protection 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	0x00000000																															
Access	RW																															
Name	SUBSECTOR																															

Bit	Name	Reset	Access	Description
31:0	SUBSECTOR	0x00000000	RW	**Lower Block Number**
	The block number that defines the lower block in the range of blocks that is to be locked from writing. The definition of a block in terms of number of bytes is programmable via the QSPIn_DEVSIZECONFIG register.			

37.5.19 QSPIn_UPPERWRPROT - Upper Write Protection 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	0x00000000																															
Access	RW																															
Name	SUBSECTOR																															

Bit	Name	Reset	Access	Description
31:0	SUBSECTOR	0x00000000	RW	**Upper Block Number**
	The block number that defines the upper block in the range of blocks that is to be locked from writing. The definition of a block in terms of number of bytes is programmable via the QSPIn_DEVSIZECONFIG register.			

37.5.20 QSPIn_WRPROTCTRL - Write Protection Control 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																															0	0
Access																															RW	RW
Name																															ENB	INV

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	ENB	0	RW	**Write Protection Enable Bit** When set to 1, any write access with an address within the protection region defined in the lower and upper write protection registers is rejected. A bus fault is generated and an interrupt source triggered. When set to 0, the protection region is disabled.
0	INV	0	RW	**Write Protection Inversion Bit** When set to 1, the protection region defined in the lower and upper write protection registers is inverted meaning it is the region that the system is permitted to write to. When set to 0, the protection region defined in the lower and upper write protection registers is the region that the system is not permitted to write to.

37.5.21 QSPIn_INDIRECTREADXFERCTRL - Indirect Read Transfer Control 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		0	0	0	0	0	0
Access																									R		RWH	R	RWH	R	W1	W1
Name																									NUMINDOPSDONE		INDOPSDONESTATUS	RDQUEUED	SRAMFULL	RDSTATUS	CANCEL	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*		
7:6	NUMINDOPSDONE	0x0	R	**Number Indirect Operations Done**
	This field contains the number of indirect operations which have been completed. This is used in conjunction with INDOPSDONESTATUS. It is incremented by hardware when an indirect operation has completed. Write a 1 to INDOPSDONESTATUS to decrement it.			
5	INDOPSDONESTATUS	0	RWH	**Indirect Completion Status**
	This field is set to 1 when an indirect operation has completed. Write a 1 to this field to clear it.			
4	RDQUEUED	0	R	**Two Indirect Read Operations Have Been Queued**
	Two indirect read operations have been queued			
3	SRAMFULL	0	RWH	**SRAM Full**
	SRAM full and unable to immediately complete an indirect operation. Write a 1 to this field to clear it.			
2	RDSTATUS	0	R	**Indirect Read Status**
	Indirect read operation in progress (status)			
1	CANCEL	0	W1	**Cancel Indirect Read**
	Writing a 1 to this bit will cancel all ongoing indirect read operations.			
0	START	0	W1	**Start Indirect Read**
	Writing a 1 to this bit will trigger an indirect read operation. The assumption is that the indirect start address and the indirect number of bytes register is setup before triggering the indirect read operation.			

37.5.22 QSPIn_INDIRECTREADXFERWATERMARK - Indirect Read Transfer Watermark 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	0x00000000																															
Access	RW																															
Name	LEVEL																															

Bit	Name	Reset	Access	Description
31:0	LEVEL	0x00000000	RW	**Watermark Value**
	When the SRAM fill level passes the watermark, an interrupt is generated. This field can be disabled by writing a value of all zeroes. Value is in unit of bytes.			

37.5.23 QSPIn_INDIRECTREADXFERSTART - Indirect Read Transfer Start Address 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	0x00000000																															
Access	RW																															
Name	ADDR																															

Bit	Name	Reset	Access	Description
31:0	ADDR	0x00000000	RW	**Indirect Read Transfer Start Address**
	This is the start address from which the indirect access will commence its READ operation.			

37.5.24 QSPIn_INDIRECTREADXFERNUMBYTES - Indirect Read Transfer Number Bytes Register

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	0x00000000																															
Access	RW																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	RW	**Indirect Read Transfer Number Bytes**
	This is the number of bytes that the indirect access will consume. This can be bigger than the size of SRAM.			

37.5.25 QSPIn_INDIRECTWRITEXFERCTRL - Indirect Write Transfer Control 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		0	0		0	0	0
Access																									R		RWH	R		R	W1	W1
Name																									NUMINDOPSDONE		INDOPSDONESTATUS	WRQUEUED		WRSTATUS	CANCEL	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*		
7:6	NUMINDOPSDONE	0x0	R	**Indirect Operations Done**
	This field contains the number of indirect operations which have been completed. This is used in conjunction with INDOPSDONESTATUS. It is incremented by hardware when an indirect operation has completed. Write a 1 to INDOPSDONESTATUS to decrement it.			
5	INDOPSDONESTATUS	0	RWH	**Indirect Completion Status**
	This field is set to 1 when an indirect operation has completed. Write a 1 to this field to clear it.			
4	WRQUEUED	0	R	**Two Indirect Write Operations Have Been Queued**
	Two indirect write operations have been queued			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	WRSTATUS	0	R	**Indirect Write Status**
	Indirect write operation in progress			
1	CANCEL	0	W1	**Cancel Indirect Write**
	Writing a 1 to this bit will cancel all ongoing indirect write operations.			
0	START	0	W1	**Start Indirect Write**
	Writing a 1 to this bit will trigger an indirect write operation. The assumption is that the indirect start address and the indirect number of bytes register is setup before triggering the indirect write operation.			

37.5.26 QSPIn_INDIRECTWRITEXFERWATERMARK - Indirect Write Transfer Watermark 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	0xFFFFFFFF																															
Access	RW																															
Name	LEVEL																															

Bit	Name	Reset	Access	Description
31:0	LEVEL	0xFFFFFFFF	RW	**Watermark Value**
	This represents the maximum fill level of the SRAM before a DMA peripheral access is permitted. When the SRAM fill level falls below the watermark, an interrupt is also generated. This field can be disabled by writing a value of all ones. Value is in unit of bytes.			

37.5.27 QSPIn_INDIRECTWRITEXFERSTART - Indirect Write Transfer Start Address 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	0x00000000																															
Access	RW																															
Name	ADDR																															

Bit	Name	Reset	Access	Description
31:0	ADDR	0x00000000	RW	**Start of Indirect Access**
	This is the start address from which the indirect access will commence its READ operation.			

37.5.28 QSPIn_INDIRECTWRITEXFERNUMBYTES - Indirect Write Transfer Number Bytes Register

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	0x00000000																															
Access	RW																															
Name	VALUE																															

Bit	Name	Reset	Access	Description
31:0	VALUE	0x00000000	RW	**Indirect Number of Bytes**
	This is the number of bytes that the indirect access will consume. This can be bigger than the configured size of SRAM.			

37.5.29 QSPIn_INDIRECTTRIGGERADDRRANGE - Indirect Trigger Address Range 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																													0x4		
Access																													RW		
Name																													INDRANGEWIDTH		

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	INDRANGEWIDTH	0x4	RW	**Indirect Trigger Address Width**
	This specifies the width of the indirect trigger address range.			

37.5.30 QSPIn_FLASHCOMMANDCTRLMEM - Flash Command Control Memory Register (STIG)

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				0x000										0x0			0x00														0	0
Access				RW										RW			R														R	W1
Name				MEMBANKADDR										NBOFSTIGREADBYTES			MEMBANKREADDATA														MEMBANKREQINPROGRESS	TRIGGERMEMBANKREQ

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*		
28:20	MEMBANKADDR	0x000	RW	**Memory Bank Address**
	The address of the Memory Bank which data will be read from.			
19	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
18:16	NBOFSTIGREADBYTES	0x0	RW	**Number of Read Bytes for the Extended STIG**
	Defines the number of read bytes for the extended STIG.			
15:8	MEMBANKREADDATA	0x00	R	**Last Requested Data From the STIG Memory Bank**
	Last requested data from the STIG Memory Bank.			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	MEMBANKREQINPROGRESS	0	R	**Memory Bank Data Request in Progress**
	Memory Bank data request in progress.			
0	TRIGGERMEMBANKREQ	0	W1	**Trigger the Memory Bank Data Request**
	Trigger the Memory Bank data request.			

37.5.31 QSPIn_FLASHCMDCTRL - Flash Command Control Register (STIG)

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	0x00								0	0x0			0	0	0x0		0	0x0			0x00									0	0	0
Access	RW								RW	RW			RW	RW	RW		RW	RW			RW									RW	R	W1
Name	CMDOPCODE								ENBREADDATA	NUMRDDATABYTES			ENBCOMDADDR	ENBMODEBIT	NUMADDRBYTES		ENBWRITEDATA	NUMWRDATABYTES			NUMDUMMYCYCLES									STIGMEMBANKEN	CMDEXECSTATUS	CMDEXEC

Bit	Name	Reset	Access	Description
31:24	CMDOPCODE	0x00	RW	**Command Opcode**
	The command opcode field should be setup before triggering the command. Writing to CMDEXEC launches the command. NOTE: The command mode depends on the INSTRTYPE field in the QSPIn_DEVINSTRRDCONFIG register.			
23	ENBREADDATA	0	RW	**Read Data Enable**
	Set to 1 if the command specified in CMDOPCODE will read data from the device.			
22:20	NUMRDDATABYTES	0x0	RW	**Number of Read Data Bytes**
	Up to 8 data bytes may be read using this command. Set to 0 for 1 byte and 7 for 8 bytes.			
19	ENBCOMDADDR	0	RW	**Command Address Enable**
	Set to 1 if the command specified in CMDOPCODE requires an address. This should be set before triggering the command with CMDEXEC			
18	ENBMODEBIT	0	RW	**Mode Bit Enable**
	Set to 1 to ensure the mode bits as defined in the QSPIn_MODEBITCONFIG register are sent following the address bytes for STIG commands.			
17:16	NUMADDRBYTES	0x0	RW	**Number of Address Bytes**
	Set to the number of address bytes required (the address itself is programmed in the QSPIn_FLASHCMDADDR). This should be setup before triggering the command with CMDEXEC. 0 : 1 address byte. 1 : 2 address bytes. 2 : 3 address bytes. 3 : 4 address bytes.			
15	ENBWRITEDATA	0	RW	**Write Data Enable**
	Set to 1 if the command specified in the command opcode field requires write data bytes to be sent to the device.			
14:12	NUMWRDATA-BYTES	0x0	RW	**Number of Write Data Bytes**
	Up to 8 Data bytes may be written using this command Set to 0 for 1 byte, 7 for 8 bytes.			
11:7	NUMDUMMYCY-CLES	0x00	RW	**Number of Dummy Cycles**
	Set to the number of dummy cycles required. This should be setup before triggering the command via CMDEXEC.			
6:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	STIGMEMBANKEN	0	RW	**STIG Memory Bank Enable Bit**
	STIG Memory Bank enable bit.			

Bit	Name	Reset	Access	Description
1	CMDEXECSTATUS	0	R	**Command Execution in Progress**
	Command execution in progress.			
0	CMDEXEC	0	W1	**Execute the Command**
	Execute the command.			

37.5.32 QSPIn_FLASHCMDADDR - Flash Command Address Register (STIG)

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	0x00000000																															
Access	RW																															
Name	ADDR																															

Bit	Name	Reset	Access	Description
31:0	ADDR	0x00000000	RW	**Command Address**
	Should be configured before triggering the STIG command. It is the address used by the command specified in CMDOPCODE of QSPIn_FLASHCMDCTRL.			

37.5.33 QSPIn_FLASHRDDATALOWER - Flash Command Read Data Register (Lower) (STIG)

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	0x00000000																															
Access	R																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	R	**Read Data Lower**
	This is the data that is returned by the flash device for any status or configuration read operation using STIG. The register will be valid when CMDEXECSTATUS is low.			

37.5.34 QSPIn_FLASHRDDATAUPPER - Flash Command Read Data Register (Upper) (STIG)

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	0x00000000																															
Access	R																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	R	**Read Data Upper**
	This is the data that is returned by the FLASH device for any status or configuration read operation using STIG. The register will be valid when CMDEXECSTATUS is low.			

37.5.35 QSPIn_FLASHWRDATALOWER - Flash Command Write Data Register (Lower) (STIG)

Offset	Bit Position																													
0x0A8	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	RW																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	RW	**Command Write Data Lower Byte**
	This is the command write data lower byte for STIG writes. This should be setup before triggering the command with CMDEXEC of the QSPIn_FLASHCMDCTRL register.			

37.5.36 QSPIn_FLASHWRDATAUPPER - Flash Command Write Data Register (Upper) (STIG)

Offset	Bit Position																													
0x0AC	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	RW																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	RW	**Command Write Data Upper Byte**
	This is the command write data upper byte for STIG writes. This should be setup before triggering the command with CMDEXEC of the QSPIn_FLASHCMDCTRL register.			

37.5.37 QSPIn_POLLINGFLASHSTATUS - Polling Flash Status Register

Offset	Bit Position																													
0x0B0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0x0											0	0x00							
Access													RW											R	R							
Name													DEVICESTATUSNBDUMMY											DEVICESTATUSVALID	DEVICESTATUS							

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	DEVICESTATUSNB-DUMMY	0x0	RW	**Auto-polling Dummy Cycles**
	Number of dummy cycles for auto-polling			
15:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	DEVICESTATUS-VALID	0	R	**Device Status Valid**
	This is set when value in DEVICESTATUS is valid.			
7:0	DEVICESTATUS	0x00	R	**Device Status**
	Status Register of Device for the latest auto-poll.			

37.5.38 QSPIn_PHYCONFIGURATION - PHY Configuration Register

Offset	Bit Position																															
0x0B4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0												0x00																0x00			
Access	W1												RW																RW			
Name	PHYCONFIGRESYNC												PHYCONFIGTXDLLDELAY																PHYCONFIGRXDLLDELAY			

Bit	Name	Reset	Access	Description
31	PHYCONFIGRE-SYNC	0	W1	**PHY Config Resync**
	This bit is used for re-synchronizing delay lines to update them with values from PHYCONFIGTXDLLDELAY and PHYCONFIGRXDLLDELAY fields.			
30:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22:16	PHYCONFIGTXDLL-DELAY	0x00	RW	**TX DLL Delay**
	This field determines the number of delay elements to insert on TX clock.			
15:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	PHYCONFIGRXDLL-DELAY	0x00	RW	**RX DLL Delay**
	This field determines the number of delay elements to insert on RX clock.			

37.5.39 QSPIn_OPCODEEXTLOWER - Opcode Extension Register (Lower)

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	0x13								0xED								0xFA								0x00							
Access	RW								RW								RW								RW							
Name	EXTREADOPCODE								EXTWRITEOPCODE								EXTPOLLOPCODE								EXTSTIGOPCODE							

Bit	Name	Reset	Access	Description
31:24	EXTREADOPCODE	0x13	RW	**Read Opcode Extension**
	Supplement byte of any Read Opcode			
23:16	EXTWRITEOPCODE	0xED	RW	**Write Opcode Extension**
	Supplement byte of any Write Opcode			
15:8	EXTPOLLOPCODE	0xFA	RW	**Polling Opcode Extension**
	Supplement byte of any Polling Opcode			
7:0	EXTSTIGOPCODE	0x00	RW	**STIG Opcode Extension**
	Supplement byte of any STIG Opcode			

37.5.40 QSPIn_OPCODEEXTUPPER - Opcode Extension Register (Upper)

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	0x06								0xF9																							
Access	RW								RW																							
Name	WELOPCODE								EXTWELOPCODE																							

Bit	Name	Reset	Access	Description
31:24	WELOPCODE	0x06	RW	**WEL Opcode**
	First byte of any WEL Opcode			
23:16	EXTWELOPCODE	0xF9	RW	**WEL Opcode Extension**
	Supplement byte of any WEL Opcode			
15:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

37.5.41 QSPIn_MODULEID - Module ID 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	0x04								0x0003																						0x0	
Access	R								R																						R	
Name	FIXPATCH								MODULEID																						CONF	

Bit	Name	Reset	Access	Description
31:24	FIXPATCH	0x04	R	**Fix/patch Number**
	Fix/patch number			
23:8	MODULEID	0x0003	R	**Module/Revision ID Number**
	Module/Revision ID number			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	CONF	0x0	R	**Configuration ID Number**
	0 : OCTAL + PHY Configuration 1 : OCTAL Configuration 2 : QUAD + PHY Configuration 3 : QUAD Configuration			

37.5.42 QSPIn_ROUTEPEN - I/O Routing Pin 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															0	0		0	0	0	0	0	0	0	0	0	0			0	0	0
Access															RW	RW		RW	RW	RW	RW	RW	RW	RW	RW	RW	RW			RW	RW	RW
Name															RST1PEN	RST0PEN		SCLKINPEN	DQSPEN	DQ7PEN	DQ6PEN	DQ5PEN	DQ4PEN	DQ3PEN	DQ2PEN	DQ1PEN	DQ0PEN			CS1PEN	CS0PEN	SCLKPEN

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	RST1PEN	0	RW	**RST1 Pin Enable**
	Pin Enable bit for RST1			
16	RST0PEN	0	RW	**RST0 Pin Enable**
	Pin Enable bit for RST0			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	SCLKINPEN	0	RW	**SCLKIN Pin Enable**
	Pin Enable bit for returned SCLK. This bit should not be set.			
13	DQSPEN	0	RW	**DQS Pin Enable**
	Pin Enable bit for DQS. The use of DQS is not recommended.			
12	DQ7PEN	0	RW	**DQ7 Pin Enable**
	Pin Enable bit for DQ7			
11	DQ6PEN	0	RW	**DQ6 Pin Enable**
	Pin Enable bit for DQ6			
10	DQ5PEN	0	RW	**DQ5 Pin Enable**
	Pin Enable bit for DQ5			
9	DQ4PEN	0	RW	**DQ4 Pin Enable**
	Pin Enable bit for DQ4			
8	DQ3PEN	0	RW	**DQ3 Pin Enable**
	Pin Enable bit for DQ3			
7	DQ2PEN	0	RW	**DQ2 Pin Enable**
	Pin Enable bit for DQ2			
6	DQ1PEN	0	RW	**DQ1 Pin Enable**
	Pin Enable bit for DQ1			
5	DQ0PEN	0	RW	**DQ0 Pin Enable**
	Pin Enable bit for DQ0			

Bit	Name	Reset	Access	Description
4:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	CS1PEN	0	RW	**CS1 Pin Enable**
	Pin Enable bit for CS1			
1	CS0PEN	0	RW	**CS0 Pin Enable**
	Pin Enable bit for CS0			
0	SCLKPEN	0	RW	**SCLK Pin Enable**
	Pin Enable bit for SCLK			

37.5.43 QSPIn_ROUTELOC0 - I/O Route Location Register 0

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																					0x00						0x00					
Access																					RW						RW					
Name																					QSPIRSTLOC						QSPILOC					

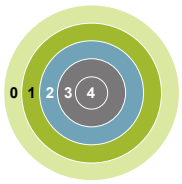
Bit	Name	Reset	Access	Description
31:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:6	QSPIRSTLOC	0x00	RW	**I/O Location**
	Decides the location of the QSPI RST0, RST1 I/O pins.			

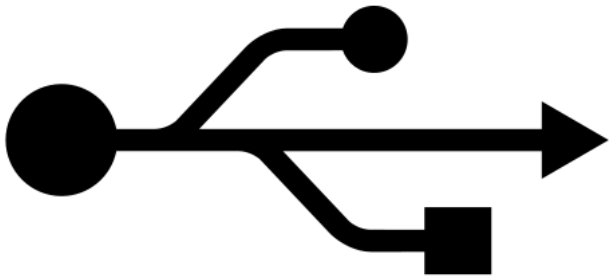
Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1

Bit	Name	Reset	Access	Description
5:0	QSPILOC	0x00	RW	**I/O Location**
	Decides the location of the QSPI I/O pins.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1

38. USB - Universal Serial Bus Controller





Quick Facts

What?

The USB is a full-speed/low-speed USB 2.0 compliant USB Controller that can be used in Device and Host configurations. The on-chip 3.3 V regulator delivers up to 200 mA and can also be used to power external components, eliminating the need for an external LDO.

Why?

USB provides a robust, industry-standard way to interface to PCs and other portable devices.

How?

The flexible and highly software-configurable architecture of the USB Controller makes it easy to implement both device- and host-capable solutions. The on-chip PHY with software controllable pull-up and pull-down resistors, VBUS comparators and ID-line detection reduces the number of external components to a minimum. Third-party USB software stacks are also available, reducing the development time substantially. By utilizing the very low energy consumption in EM2, the USB device will be able to wake up and perform tasks several times a second without violating the 2.5 mA maximum average current during suspend.

38.1 Introduction

The USB is a full-speed/low-speed USB 2.0 compliant host/device controller. The architecture is very flexible and allows the USB to be used in Device and Host-only configurations. The on-chip voltage regulator and PHY minimizes the number of external components. A switchable external 5V supply or step-up regulator is needed for Host configurations.

38.2 Features

- Fully compliant with Universal Serial Bus Specification, Revision 2.0
- Supports full-speed (12 MBit/s) and low-speed (1.5 MBit/s) host and device
- Low Energy Mode, reducing average current consumption by up to 90%
- Dedicated Internal DMA Controller
- 12 software-configurable endpoints (6 IN, 6 OUT) in addition to endpoint 0
- 2 KB endpoint memory
- Resume/Reset detection in EM2 (during suspend)
- Session Request Protocol (SRP) detection in EM2 (during host session off) using data-line pulsing only (VBUS pulsing not supported)
- Soft connect/disconnect
- Charger detection circuitry with automatic detection of SDP, CDP, and DCP interfaces.
- D+ and D- can be routed to ADC input to support ACM and proprietary charger architectures.
- On-chip PHY
  - Internal pull-up and pull-down resistors
  - Voltage comparators for monitoring VBUS voltage
  - A/B Device identification using ID line
- Internal 5V to 3.3V Regulator
  - Output voltage range: 2.4V to 3.8V (3.3V required for USB operation)
  - Output current: 200 mA
  - Input voltage range: 2.7 - 5.5V
  - Enabled automatically when input voltage applied
  - Low quiescent current: 9 uA
  - Dedicated input pin allows regulator to be used in host configurations
  - Regulator output pin can be used to power the EFM32 Giant Gecko 12, as well as external components
  - Regulator voltage output sense feature for detecting USB plug/unplug events (also available in EM2/3)

38.3 USB System Description

A block diagram of the USB is shown in Figure 38.1 USB Block Diagram on page 1441.

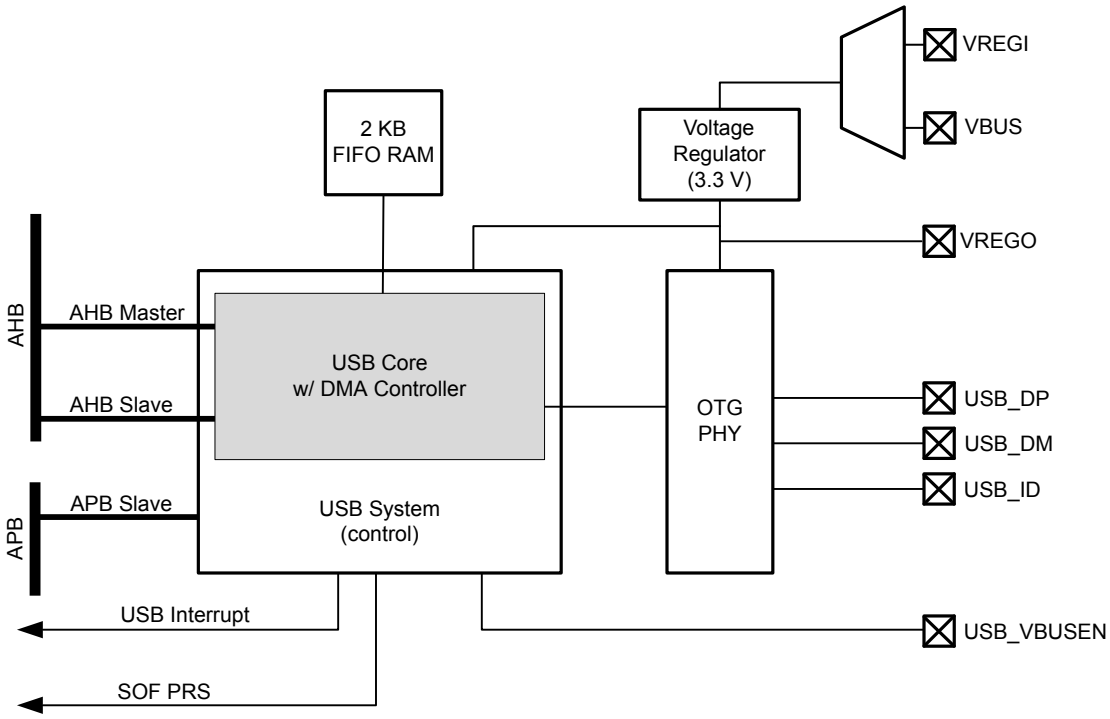


Figure 38.1. USB Block Diagram

The USB consists of digital logic, a 2 KB endpoint RAM, PHY and a voltage regulator with output voltage sensor. The voltage regulator provides a stable 3.3 V supply for the PHY, but can also be used to power the EFM32 Giant Gecko 12 itself as well as external components.

The digital logic of the USB is split into three blocks: USB System, USB Core and USB Charger Detection.

The USB System block is accessed using USB registers from offset 0x000 to 0x03C and controls the voltage regulator, low energy mode, and enabling/disabling of the PHY and USB pins. USB System is clocked by $HFBUSCLK_{USB}$ and is accessed using an APB slave interface.

The USB core block is clocked by the USB Core Clock (USBCLK) and is accessed using an AHB slave interface. This interface is used for accessing the FIFO contents and the registers in the core part starting at offset 0xDE000. An additional master interface is used by the internal DMA controller of the core. The core part takes care of all the USB protocol related functionality. The USBCLK must not be disabled when the core part is active - in order to access the USB system registers, the USBCLK must be enabled.

The USB Charger Detection block is clocked by $HFCORECLK_{USB}$ and ULFRCOCLK, and is controlled by the USB System block. This block manages the charger detection related functionality specified by the USB Battery Charging Specification.

38.3.1 USB Pins

There are several pins associated with the USB.

- USB_DP (USB D+) and USB_DM (USB D-) are the USB data signaling pins.
- USB_ID is the ID pin used to detect the USB device type (A or B). This pin can be left unconnected when not used.
- VBUS has multiple functions, and should typically be connected to the USB 5V (VBUS) pin on the USB receptacle. Internally, it is connected to the voltage comparators and current sink/source in the PHY. It can also be the input supply to the 3.3 V Regulator (determined by the INPUTMODE bitfield in the EMU_R5VCTRL register). This pin should (typically) have at minimum a 1.0 uF capacitor. If unused, the VBUS input may be left floating - a weak internal pull-down will ensure this pin remains at ground.
- VREGI is a second input supply for the 3.3 V regulator and (determined by the INPUTMODE bitfield in the EMU_R5VCTRL register). If used as a regulator supply input, this pin should have a 1.0 uF capacitor. If unused, the VREGI input may be left floating - a weak internal pull-down will ensure this pin remains at ground.
- VREGO is the 3.3 V regulator output. This pin should have a 4.7 uF output capacitor.
- USB_VBUSEN is used to turn on and off power to USB VBUS when operating as a USB Host. USB_VBUSEN will be high-impedance until the pins are enabled from software. Thus, if a defined level is required during startup an external pull-up/pull-down can be used.

38.3.2 USB Initialization

Follow these steps to enable the USB:

1. Enable the clock to the system part by setting USB in CMU_HFBUSCLKEN0, and enable the USBCLK by setting USBCLKEN and USBCLKSEL in the CMU_USBCTRL register.
2. If the internal USB regulator is bypassed (by applying 3.3V on VREGI/VBUS and VREGO externally), disable the regulator by setting the BYPASS bit in the EMU_R5VCTRL register.
3. If the internal USB regulator is used, before enabling the USB PHY firmware should ensure that the regulator output voltage is 3.3V by programming the proper value to EMU_R5VOUTLEVEL, and then waiting for the EMU_R5VVSINT interrupt.
4. Enable the USB PHY pins by setting PHYPEN in USB_ROUTE.
5. For a USB Host , set VBUSENAP in USB_CTRL to the desired value and then enable the USB_VBUSEN pin in USB_ROUTE. Set the MODE for the pin to PUSHPULL.
6. Wait for the core to come out of reset. The simplest method is by polling a core register with non-zero reset value until it reads a non-zero value. This takes approximately twenty 48-MHz cycles.
7. Begin initializing the USB core as described in USB Core Description.

38.3.3 Configurations

The USB can be used as Device or Host. The sections below describe the different configurations. External ESD protection and series resistors for impedance matching are required. The voltage regulator requires a 1.0 uF external decoupling capacitor on the input and a 4.7 uF external decoupling capacitor on the output. Decoupling not related to USB is not shown in the figures.

In USB host mode, an external 48 MHz or 24 MHz xtal (2500ppm or better) is required. For USB device mode, USB may use an external crystal or it may be clocked from its own internal oscillator.

38.3.3.1 Bus-powered Device

Bus-powered configuration is enabled by setting the SELFPOWERED bit to 0 in USB_CTRL. A bus-powered device configuration is shown in Figure 38.2 Bus-powered Device on page 1443. In this configuration, the USB 5V supply powers the internal 3.3 V voltage regulator to power the USB PHY and the EFM32 Giant Gecko 12 at 3.3 V. The voltage regulator output (VREGO) may also be used to power other components of the system.

If unused, the VREGI input may be left floating - a weak internal pull-down will ensure this pin remains at ground.

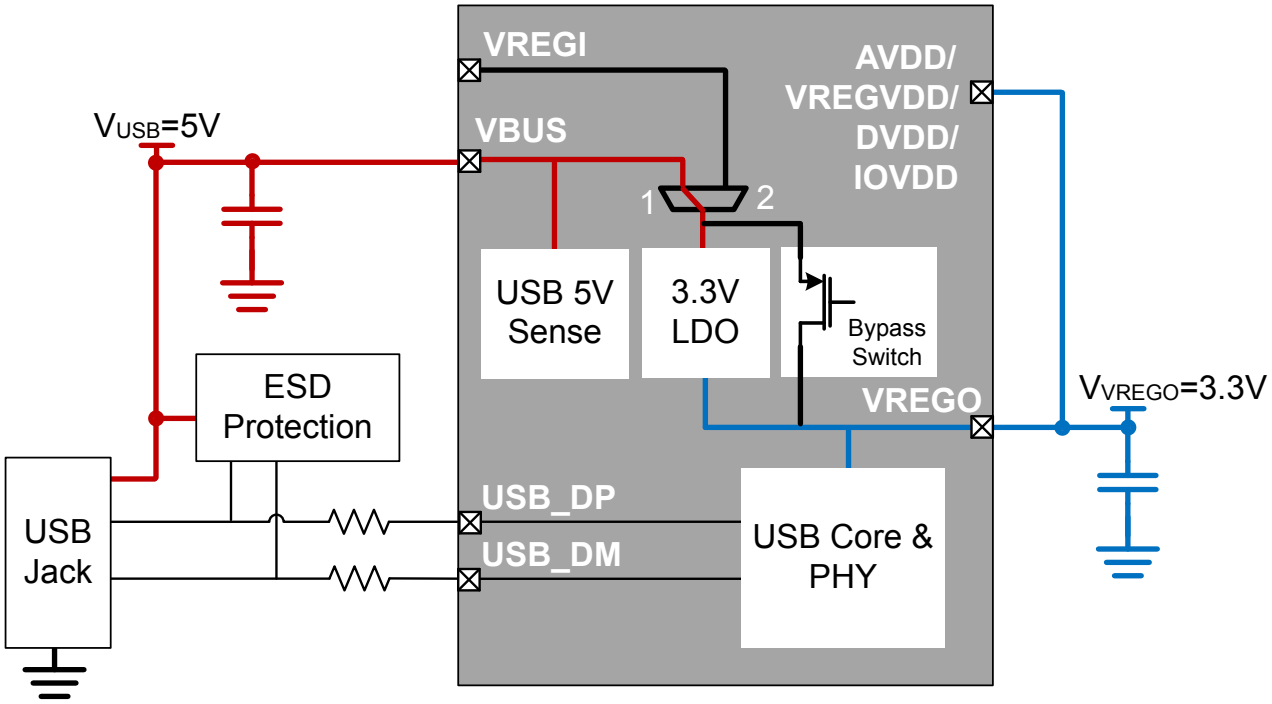


Figure 38.2. Bus-powered Device

38.3.3.2 Self-powered Device

Self-powered configuration is enabled by setting the SELFPOWERED bit to 1 in USB_CTRL. A self-powered device configuration is shown in Figure 38.3 Self-powered Device on page 1444. When the USB is configured as a self-powered device, the voltage regulator is typically used to power the PHY only, allthough it may also be used to power other external components. When the USB is connected to a host, the interal 3.3 V voltage regulator is activated. Software can detect a USB connection event by enabling the VBUS detect interrupt high (i.e., by setting VBUSDETH in USB_IEN). The PHY pins can then be enabled and USB traffic can begin. The VBUS detect interrupt low can be used to detect when the USB 5V voltage disappears (e.g., if the USB cable is unplugged).

If unused, the VREGI input may be left floating - a weak internal pull-down will ensure this pin remains at ground.

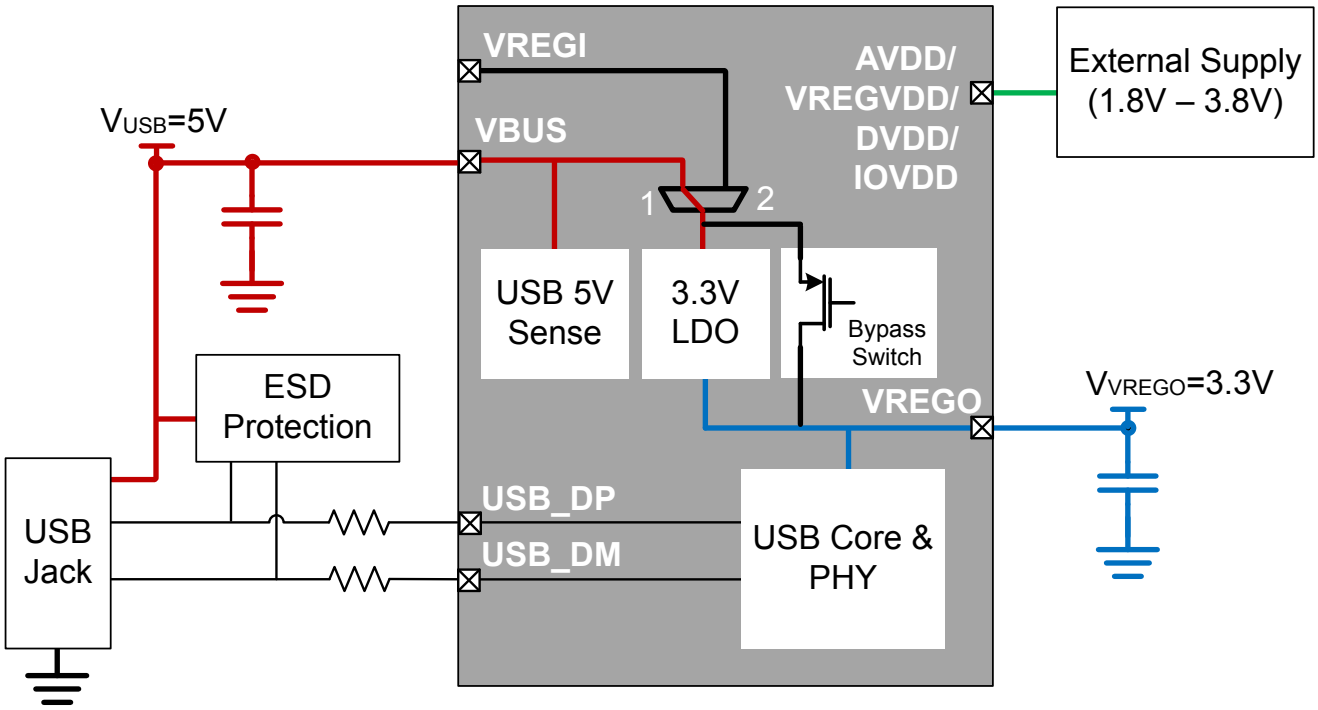


Figure 38.3. Self-powered Device

38.3.3.3 Dual-powered Device (with Internal Switch)

A battery- or external-powered device may switch its power supply to the USB 5V supply when connected to a USB host. This is typically useful for extending the life of battery-powered devices. An internal switch (controlled by EMU.R5VCTRL.INPUTMODE) allows the voltage regulator supply input to seamlessly switch between a battery (or other external supply) and the USB 5V supply.

Typically, firmware would set EMU.R5VCTRL.INPUTMODE = AUTO to allow the 3.3V LDO to be sourced from the higher of the two supply inputs (VREGI or VBUS).

Alternately, the VBUS high detection interrupt could be used to detect the presence of USB 5V. Firmware can then set EMU.R5VCTRL.INPUTMODE = VBUS. If necessary, the application may have to reduce the current consumption before switching to the USB power source to avoid exceeding USB current limits. In this scenario, firmware should also enable the VBUS low detection interrupt to allow firmware to respond quickly when the USB 5V voltage is removed. For example, when USB is disconnected, firmware may reduce the current consumption (e.g., by reducing the clock frequency) to avoid excessive voltage droop on the VREGO supply, and also set EMU.R5VCTRL.INPUTMODE = VREGI to force the 3.3V LDO supply to the external supply.

This configuration is shown in Figure 38.4 Self-powered Device (with Internal Power Switch) on page 1445.

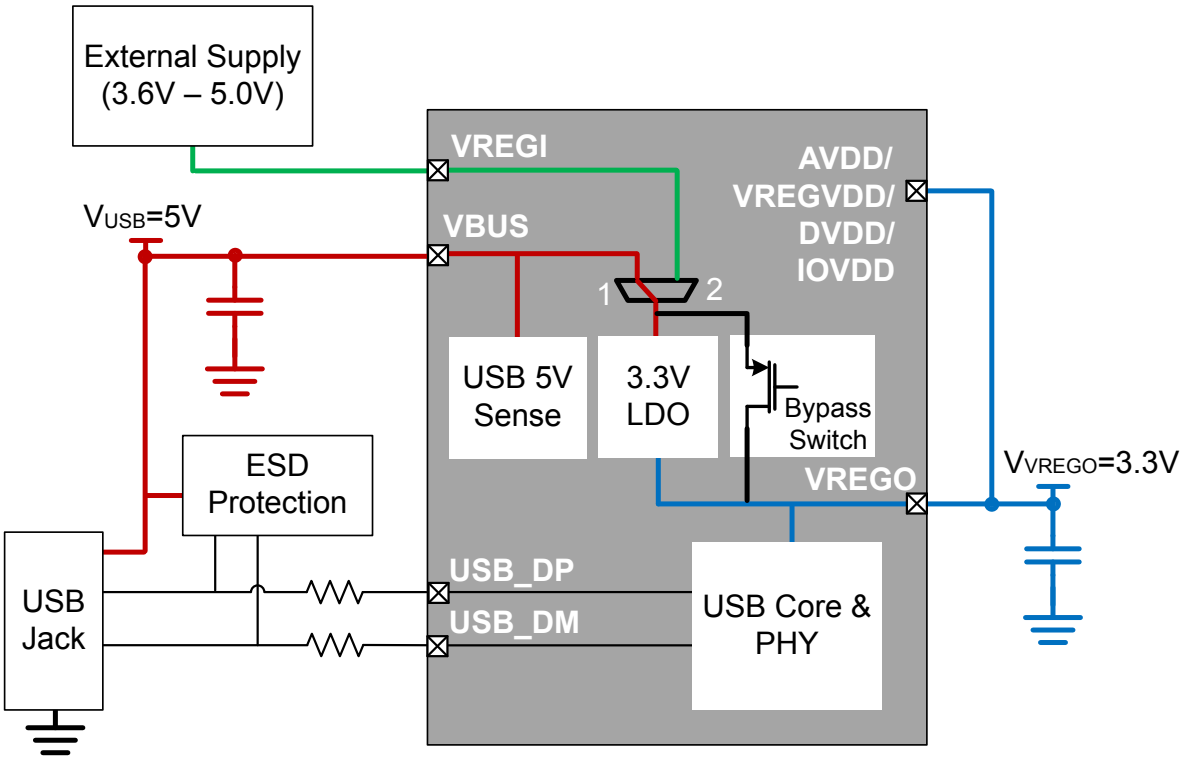


Figure 38.4. Self-powered Device (with Internal Power Switch)

38.3.3.4 Host

A host configuration is shown in Figure 38.5 Host on page 1446. In this example a 5V step-up regulator is used. In host mode, the minimum USB 5V decoupling capacitance is 96 uF. In Host configuration, the internal 3.3V LDO is unused, and the VREGO pin is driven directly from an external source. The VBUS input is still usd for USB 5V detection. If unused, the VREGI input may be left floating - a weak internal pull-down will ensure this pin remains at ground.

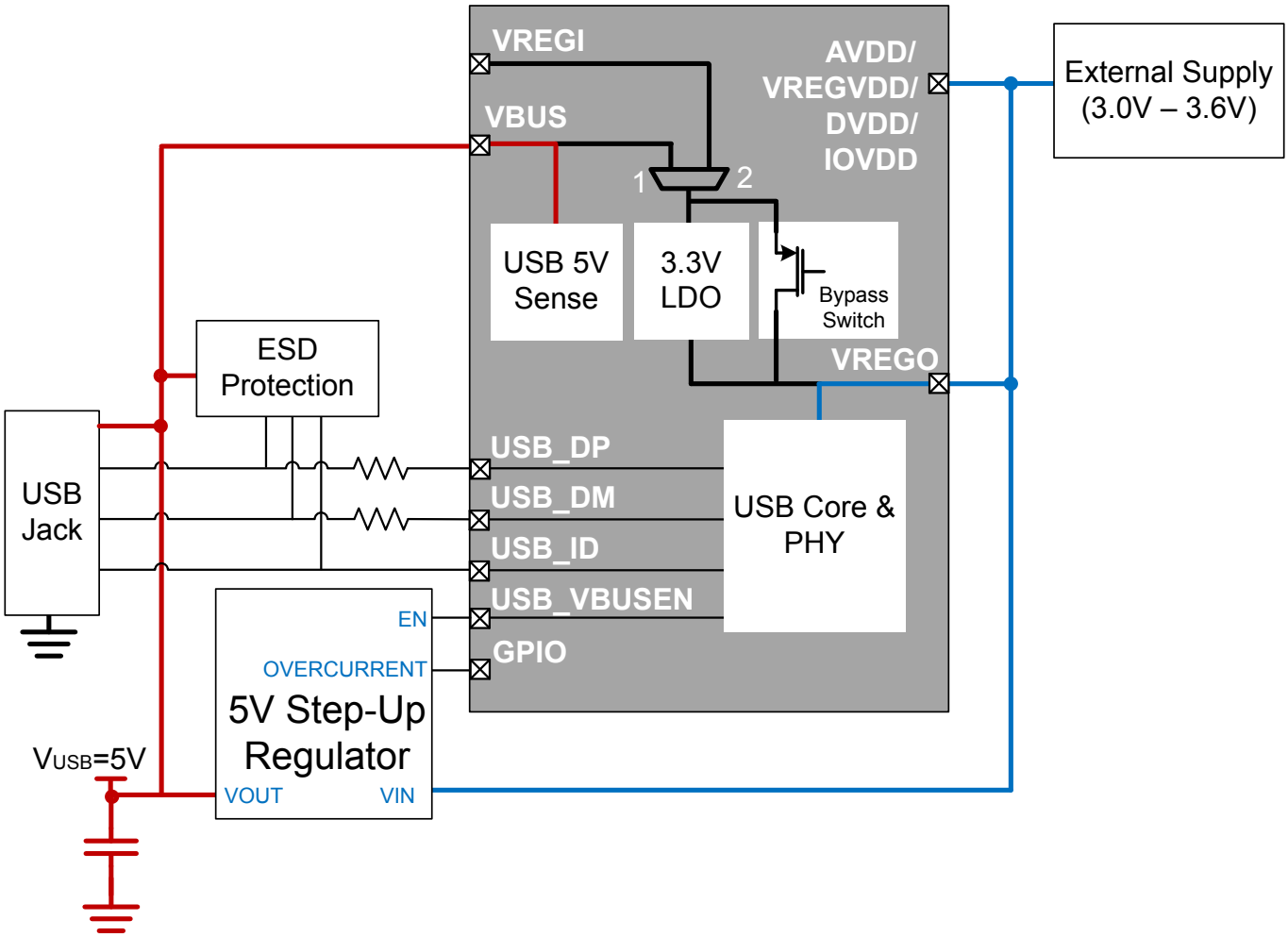


Figure 38.5. Host

38.3.4 PHY

The USB includes an internal full-speed/low-speed PHY with built-in pull-up/pull-down resistors, VBUS comparators and ID line state sensing. During suspend, the PHY enters a low-power state where only the single-ended receivers are active. The PHY is disabled by default and should be enabled by setting PHYPEN in USB_ROUTE before the USBCLK is enabled.

The PHY is powered by the internal voltage regulator output (VREGO). To power the PHY directly from an external source (for example an external 3.3 V LDO), connect both VREGO and VREGI to the external 3.3 V supply voltage. To stop the quiescent current present with the voltage regulator enabled in this configuration, disable the the regulator by setting VREGDIS in USB_CTRL after power up. Then the regulator is effectively bypassed.

When VREGO Sense is enabled, the PHY is automatically disabled internally when the VREGO Sense output is low. This will happen if VBUS-power disappears. The application can detect this by keeping the VREGO Sense Low Interrupt enabled. Note that PHYPEN in USB_ROUTE will not be set to 0 in this case.

38.3.5 Voltage Regulator

The voltage regulator is used to regulate the 5 V VBUS voltage down to 3.3 V which is the operating voltage for the PHY.

A decoupling capacitor is required on the regulator input (either VREGI or VBUS) as well as its output (VREGO). Note that the USB standard requires the total capacitance on VBUS to be between 1 uF minimum and 10 uF maximum for regular devices.

The voltage regulator is enabled by default and can thus be used to power the EFM32 Giant Gecko 12 itself. Systems not using the USB should disable the regulator by setting VREGDIS in USB_CTRL. A voltage sense circuit monitors the output voltage and can be used to detect when the voltage regulator becomes active. This sense circuit can also be used to detect when the voltage drops (e.g., when the USB cable is disconnected). If regulator voltage monitoring is not required (e.g., the VREGO voltage is always present), the sense circuit should be left disabled.

38.3.6 Interrupts and PRS

Interrupts from the core and system part share a common USB interrupt line to the CPU. The interrupt flags for the system part are grouped together in the USB_IF register. The interrupt events from the core are controlled by several core interrupt flag registers.

There are two PRS outputs from the USB: SOF and SOFSR. In Host mode, SOF toggles every time an SOF is generated. In Device mode, SOF toggles every time an SOF token is received from the USB host or when an SOF token is missed at the start of frame. In Host mode, SOFSR toggles every time an SOF is successfully transmitted. In Device mode, SOFSR toggles only when a valid SOF token is received from the USB host. Both PRS outputs must be synchronized in the PRS when used (i.e. it is an asynchronous PRS output). The edge-to-pulse converter in the PRS can be used to convert the edges into pulses if needed. The PRS outputs go to 0 in EM2/3.

38.3.7 USB Low Energy Mode

The USB also features a Low Energy Mode (LEM) that can be used to reduce the current consumption of the USB when there is no data traffic on the lines that can be received. When such a condition is detected the USB can be configured to place the PHY into a low-energy state, turn off the clock to the USB core, and potentially suspend the USHFRCO. Note that if the system is also running off of the USHFRCO, the oscillator will not be suspended when an idle condition is detected.

Low Energy Mode is enabled with the USB_CTRL_LEMIDLEEN bit. Prior to enabling Low Energy Mode, software should also configure the TIMEBASE field in USB_LEMCTRL for a 3 ms period.

Different levels of power savings are controlled by the LEMOSCCTRL and LEMPHYCTRL bitfields in the USB_CTRL register. Even though most of the USB operation is identical in Low Energy Mode, there are subtle functional differences:

1. If LEMOSCCTRL in USB_CTRL is set to SUSPEND, and the USHFRCO is turned off, it will take additional time for the CPU to access USB core registers. This is due to the fact that the system needs to restart the USHFRCO to complete any bus access. If the application will have several bus USB transactions in a short time, e.g. in the IRQ handler, it is recommended to set the LE-MOSCCTRL to GATE during those accesses.
2. Missed SOFs will not generate interrupts or toggle PRS signals when Low Energy Mode is enabled.

38.3.8 USB in EM2

During suspend and session-off EM2 should be used to save power and meet the average current requirements dictated by the USB standard. Before entering EM2, the USBCLK must be switched from 48 MHz to 32 kHz (either the LFXO or LFRCO). This is done using the CMU_USBCTRL and CMU_STATUS registers. While the USBCLK is 32 kHz, the USB core registers (starting from offset 0xDE000) cannot be accessed and the the internal DMA in the USB core will not be able to access the AHB bus. Upon EM2 wake-up, the USBCLK must be switched back to 48 MHz before accessing the core registers. The device always starts up from HFRCO so if an external crystal is used, software must restart the HFXO and switch the clock source from HFRCO to HFXO. The USB system clock, HFBUSCLK, must be kept enabled during EM2. The USB system registers can be accessed immediately upon EM2 wake-up, while running from HFRCO. Follow the steps outlined the USB Core Description when entering EM2 during suspend and session-off.

The FIFO content is lost when entering EM2. In addition, most of the USB core registers are reset and therefore need to be backed up in RAM.

In EM4, the voltage regulator is always enabled. If voltage is applied on the VREGI or VBUS pins in EM4, the regulator will draw current.

38.3.9 USB in EM3

EM3 cannot be used when the USB is active. However, EM3 can be used while waiting for the internal voltage regulator to be activated (i.e., while waiting for VBUS to reach 5V).

The FIFO RAM will be powered down and its contents lost during EM3. The application is responsible for emptying the FIFO before entering EM3.

38.3.10 USB in EM4

USB cannot be used in EM4. However, in EM4, the voltage regulator is always enabled. If a voltage is applied to the VREGI pin in EM4, the regulator will draw current.

38.4 USB Core Description

This section describes the programming requirements for the USB Core in Host and Device modes.

Important features/parameters for the core are:

- Internal DMA (Buffer Pointer Based)
- Dedicated TX FIFOS for each endpoint in device mode
- 6 IN/OUT endpoints in addition to endpoint 0 (in device mode)
- 14 host channels (in host mode)
- Dynamic FIFO sizing
- Non-Periodic Request Queue Depth: 8
- Host Mode Periodic Request Queue Depth: 8

The core has the following limitations:

- Link Power Management (LPM) is not supported
- Attached Detection Protocol (ADP) is not supported

38.4.1 Overview: Programming the Core

Each significant programming feature of the core is discussed in a separate section.

This chapter uses abbreviations for register names and their fields. For detailed information on registers, see 38.7 Register Description.

The application must perform a core initialization sequence. If the cable is connected during power-up, the Current Mode of Operation bit in the Core Interrupt register (USB_GINTSTS.CURMOD) reflects the mode. The core enters Host mode when an “A” plug is connected, or Device mode when a “B” plug is connected.

This section explains the initialization of the core after power-on. The application must follow the initialization sequence irrespective of Host or Device mode operation. All core global registers are initialized according to the core’s configuration.

1. Program the following fields in the Global AHB Configuration (USB_GAHBCFG) register.
   - DMA Mode bit
   - AHB Burst Length field
   - Global Interrupt Mask bit = 1
   - Non-periodic TxFIFO Empty Level (can be enabled only when the core is operating in Slave mode as a host.)
   - Periodic TxFIFO Empty Level (can be enabled only when the core is operating in Slave mode)
2. Program the following field in the Global Interrupt Mask (USB_GINTMSK) register:
   - USB_GINTMSK.RXFLVLMSK = 0
3. Program the following fields in USB_GUSBCFG register.
   - HNP Capable bit
   - SRP Capable bit
   - External High-Speed PHY or Internal Full-Speed Serial PHY Selection bit
   - Time-Out Calibration field
   - USB Turnaround Time field
4. The software must unmask the following bits in the USB_GINTMSK register.
   - OTG Interrupt Mask
   - Mode Mismatch Interrupt Mask
5. The software can read the USB_GINTSTS.CURMOD bit to determine whether the core is operating in Host or Device mode. The software the follows either the 38.4.1.1 Host Initialization or 38.4.1.2 Device Initialization sequence.

Note:

- The core is designed to be interrupt-driven. Polling interrupt mechanism is not recommended: this may result in undefined resolutions.
- In device mode, just after Power On Reset or a Soft Reset, the USB_GINTSTS.SOF bit is set to 1 for debug purposes. This status must be cleared and can be ignored.

38.4.1.1 Host Initialization

To initialize the core as host, the application must perform the following steps.

1. Program USB_GINTMSK.PRTINT to unmask.
2. Program the USB_HCFG register to select full-speed host.
3. Program the USB_HPRT.PRTPWR bit to 1. This drives VBUS on the USB.
4. Wait for the USB_HPRT.PRTCONNDET interrupt. This indicates that a device is connect to the port.
5. Program the USB_HPRT.PRTRST bit to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the USB_HPRT.PRTRST bit to 0.
8. Wait for the USB_HPRT.PRTENCHNG interrupt.
9. Read the USB_HPRT.PRTSPD field to get the enumerated speed.
10. Program the USB_HFIR register with a value corresponding to the selected PHY clock. At this point, the host is up and running and the port register begins to report device disconnects, etc. The port is active with SOFs occurring down the enabled port.
11. Program the RXFSIZE register to select the size of the receive FIFO.
12. Program the NPTXFSIZE register to select the size and the start address of the Non-periodic Transmit FIFO for non-periodic transactions.
13. Program the USB_HPTXFSIZ register to select the size and start address of the Periodic Transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel as described in 38.4.1.2 Device Initialization.

38.4.1.1.1 Host Connection

The following steps explain the host connection flow:

1. When the USB Cable is plugged to the Host port, the core triggers USB_GINTSTS.CONIDSTSCHNG interrupt.
2. When the Host application detects USB_GINTSTS.CONIDSTSCHNG interrupt, the application can perform one of the following actions:
   - Turn on VBUS by setting USB_HPRT.PRTPWR = 1 or
   - Wait for SRP Signaling from Device to turn on VBUS.
3. The PHY indicates VBUS power-on by detecting a VBUS valid voltage level.
4. When the Host Core detects the device connection, it triggers the Host Port Interrupt (USB_GINTSTS.PRTINT) to the application.
5. When USB_GINTSTS.PRTINT is triggered, the application reads the USB_HPRT register to check if the Port Connect Detected (USB_HPRT.PRTCONNDET) bit is set or not.

38.4.1.1.2 Host Disconnection

The following steps explain the host disconnection flow:

1. When the Device is disconnected from the USB Cable (but the cable is still connected to the USB host), the Core triggers USB_GINTSTS.DISCONNINT (Disconnect Detected) interrupt.

   Note: If the USB cable is disconnected from the Host port without removing the device, the core generates an additional interrupt - USB_GINTSTS.CONIDSTSCHNG (Connector ID Status Change).

2. The Host application can choose to turn off the VBUS by programming USB_HPRT.PRTPWR = 0.

38.4.1.2 Device Initialization

The application must perform the following steps to initialize the core at device on, power on, or after a mode change from Host to Device.

1. Program the following fields in USB_DCFG register.
   - Device Speed
   - NonZero Length Status OUT Handshake
   - Periodic Frame Interval
2. Program the Device threshold control register. This is required only if you are using DMA mode and you are planning to enable thresholding.
3. Program the USB_GINTMSK register to unmask the following interrupts.
   - USB Reset
   - Enumeration Done
   - Early Suspend
   - USB Suspend
4. Wait for the USB_GINTSTS.USBRST interrupt, which indicates a reset has been detected on the USB and lasts for about 10 ms. On receiving this interrupt, the application must perform the steps listed in 38.4.4.1.1 Initialization on USB Reset
5. Wait for the USB_GINTSTS.ENUMDONE interrupt. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the USB_DSTS register to determine the enumeration speed and perform the steps listed in 38.4.4.1.2 Initialization on Enumeration Completion

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

38.4.1.2.1 Device Connection

The device connect process varies depending on the if the VBUS is on or off when the device is connected to the USB cable.

If VBUS is on When the Device is Connected

If VBUS is on when the device is connected to the USB cable, there is no SRP from the device. The device connection flow is as follows:

1. The device triggers the USB_GINTSTS.SESSREQINT [bit 30] interrupt bit.
2. When the device application detects the USB_GINTSTS.SESSREQINT interrupt, it programs the required bits in the USB_DCFG register.
3. When the Host drives Reset, the Device triggers USB_GINTSTS.USBRST [bit 12] on detecting the Reset. The host then follows the USB 2.0 Enumeration sequence.

If VBUS is off When the Device is Connected

If VBUS is off when the device is connected to the USB cable, the device initiates SRP . The device connection flow is as follows:

1. The application initiates SRP by writing the Session Request bit in the OTG Control and Status register.
2. The host starts a new session by turning on VBUS, indicating SRP success. The core interrupts the application by setting the Session Request Success Status Change bit in the OTG Interrupt Status register.
3. The application reads the Session Request Success bit in the OTG Control and Status register and programs the required bits in USB_DCFG register.
4. When Host drives Reset, the Device triggers USB_GINTSTS.USBRST on detecting the Reset. The host then follows the USB 2.0 Enumeration sequence.

38.4.1.2.2 Device Disconnection

The device session ends when the USB cable is disconnected or if the VBUS is switched off by the Host.

The device disconnect flow is as follows:

1. When the USB cable is unplugged or when the VBUS is switched off by the Host, the Device core trigger USB_GINTSTS.OTGINT [bit 2] interrupt bit.
2. When the device application detects USB_GINTSTS.OTGINT interrupt, it checks that the USB_GOTGINT.SESENDDET (Session End Detected) bit is set to 1.

38.4.1.2.3 Device Soft Disconnection

The application can perform a soft disconnect by setting the Soft disconnect bit (SFTDISCON) in Device Control Register (USB_DCTL).

Send/Receive USB Transfers -> Soft disconnect->Soft reset->USB Device Enumeration

Sequence of operations:

1. The application configures the device to send or receive transfers.
2. The application sets the Soft disconnect bit (SFTDISCON) in the Device Control Register (USB_DCTL).
3. The application sets the Soft Reset bit (CSFTRST) in the Reset Register (USB_GRSTCTL).
4. Poll the USB_GRSTCTL register until the core clears the soft reset bit, which ensures the soft reset is completed properly.
5. Initialize the core according to the instructions in 38.4.1.2 Device Initialization.

Suspend-> Soft disconnect->Soft reset->USB Device Enumeration

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core puts the PHY in suspend mode, and the PHY clock stops.
3. The application clears the Stop PHY Clock bit in the Power and Clock Gating Control register, and waits for the PHY clock to come back. The core takes the PHY back to normal mode, and the PHY clock comes back.
4. The application sets the Soft disconnect bit (SFTDISCON) in Device Control Register (USB_DCTL).
5. The application sets the Soft Reset bit (CSFTRST) in the Reset Register (USB_GRSTCTL).
6. Poll the USB_GRSTCTL register until the core clears the soft reset bit, which ensures the soft reset is completed properly.
7. Initialize the core according to the instructions in 38.4.1.2 Device Initialization.

38.4.2 Modes of Operation

- 38.4.2.1 Overview: DMA/Slave Modes
- 38.4.2.2 DMA Mode
- 38.4.2.3 Slave Mode
- 38.4.2.4 Thresholding in DMA Mode

38.4.2.1 Overview: DMA/Slave Modes

The application can operate the core in either of two modes:

- In 38.4.2.2 DMA Mode - The core fetches the data to be transmitted or updates the received data on the AHB.
- In 38.4.2.3 Slave Mode — The application initiates the data transfers for data fetch and store.

38.4.2.2 DMA Mode

In DMA Mode, the host uses the AHB master Interface for transmit packet data fetch (AHB to USB) and receive data update (USB to AHB). The AHB master uses the programmed DMA address (USB_HCx_DMAADDR register in host mode and USB_DIEPx_DMAADDR/USB_DOEPx_DMAADDR register in device mode) to access the data buffers.

38.4.2.2.1 Transfer-Level Operation

In DMA mode, the application is interrupted only after the programmed transfer size is transmitted or received (provided the core detects no NAK/Timeout/Error response in Host mode, or Timeout/CRC Error in Device mode). The application must handle all transaction errors. In Device mode, all the USB errors are handled by the core itself.

38.4.2.2.2 Transaction-Level Operation

This mode is similar to transfer-level operation with the programmed transfer size equal to one packet size (either maximum packet size, or a short packet size).

38.4.2.3 Slave Mode

In Slave mode, the application can operate the core either in transaction-level (packet-level) operation or in pipelined transaction-level operation.

Host Mode For an OUT transaction, the application enables the channel and writes the data packet into the corresponding (Periodic or Non-periodic) transmit FIFO. The core automatically writes the channel number into the corresponding (Periodic or Non-periodic) Request Queue, along with the last DWORD write of the packet. For an IN transaction, the application enables the channel and the core automatically writes the channel number into the corresponding Request queue. The application must wait for the packet received interrupt, then empty the packet from the receive FIFO.

Device Mode For an IN transaction, the application enables the endpoint, writes the data packet into the corresponding transmit FIFO, and waits for the packet completion interrupt from the core. For an OUT transaction, the application enables the endpoint, waits for the packet received interrupt from the core, then empties the packet from the receive FIFO.

Note: The application has to finish writing one complete packet before switching to a different channel/endpoint FIFO. Violating this rule results in an error.

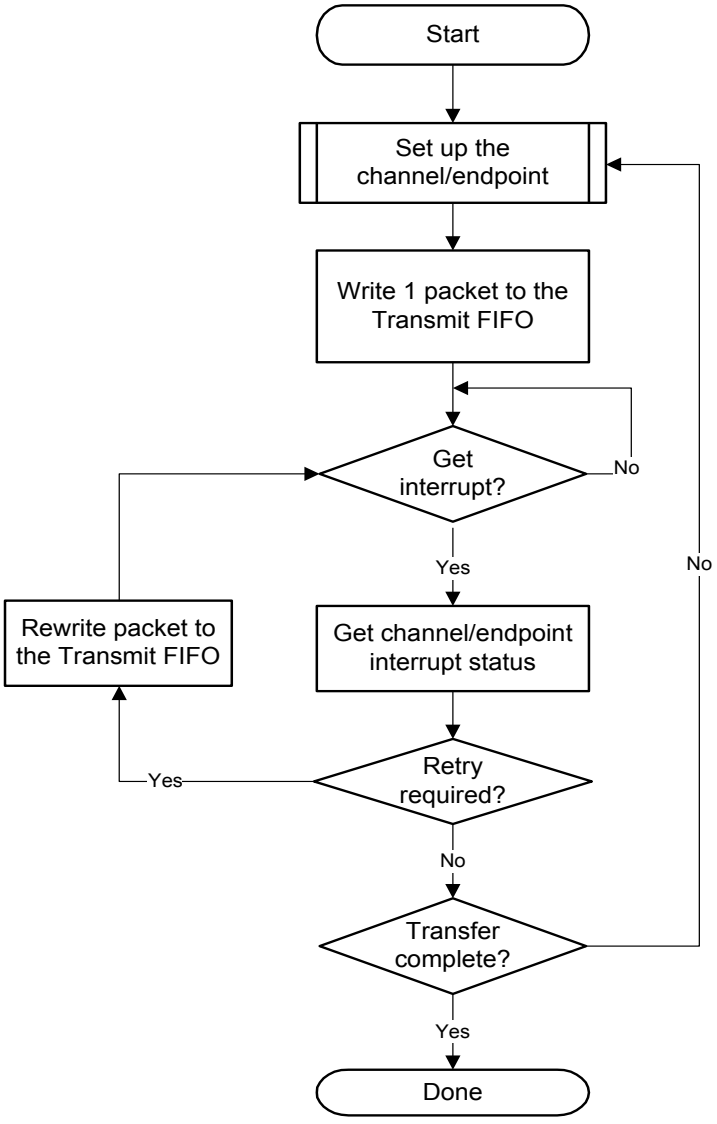


Figure 38.6. Transmit Transaction-Level Operation in Slave Mode

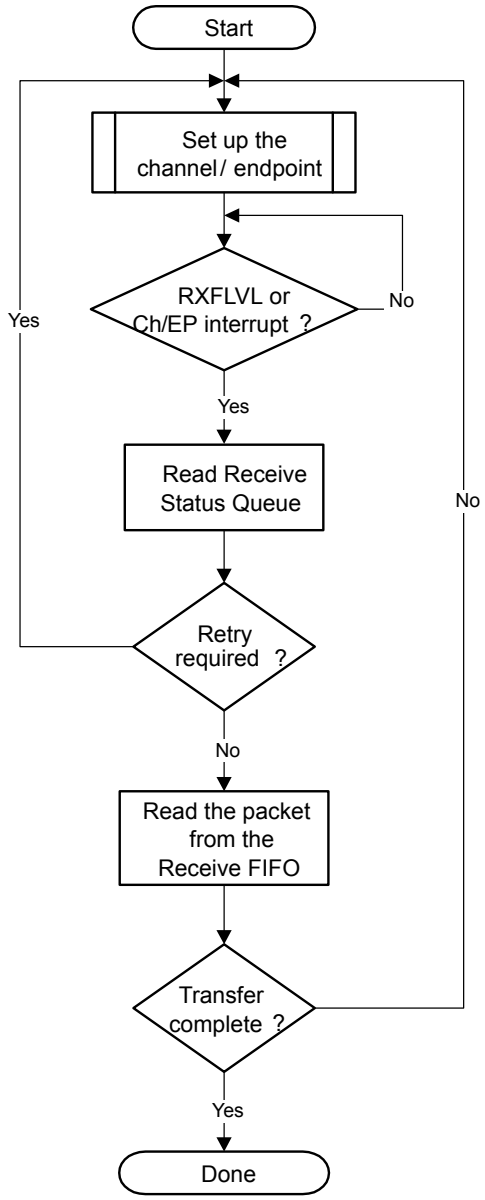


Figure 38.7. Receive Transaction-Level Operation in Slave Mode

38.4.2.3.1 Transaction-Level Operation

The application handles one data packet at a time per channel/endpoint in transaction-level operations. Based on the handshake response received on the USB, the application determines whether to retry the transaction or proceed with the next, until the end of the transfer. The application is interrupted on completion of every packet. The application performs transaction-level operations for a channel/endpoint for a transmission (host: OUT/device: IN) or reception (host: IN/device: OUT) as shown in Figure 38.6 Transmit Transaction-Level Operation in Slave Mode on page 1453 and Figure 38.7 Receive Transaction-Level Operation in Slave Mode on page 1454.

38.4.2.3.2 Pipelined Transaction-Level Operation

The application can pipeline more than one transaction (IN or OUT) with pipelined transaction-level operation, which is analogous to Transfer mode in DMA mode. In pipelined transaction-level operation, the application can program the core to perform multiple transactions. The advantage of this mode compared to transaction-level operation is that the application is not interrupted on a packet basis.

38.4.2.3.2.1 Host Mode

For an OUT transaction, the application sets up a transfer and enables the channel. The application can write multiple packets back-to-back for the same channel into the transmit FIFO, based on the space availability. It can also pipeline OUT transactions for multiple channels by writing into the HCHARn register, followed by a packet write to that channel. The core writes the channel number, along with the last DWORD write for the packet, into the Request queue and schedules transactions on the USB in the same order.

For an IN transaction, the application sets up a transfer and enables the channel, and the core writes the channel number into the Request queue. The application can schedule IN transactions on multiple channels, provided space is available in the Request queue. The core initiates an IN token on the USB only when there is enough space to receive at least of one maximum-packet-size packet of the channel in the top of the Request queue.

38.4.2.3.2.2 Device Mode

For an IN transaction, the application sets up a transfer and enables the endpoint. The application can write multiple packets back-to-back for the same endpoint into the transmit FIFO, based on available space. It can also pipeline IN transactions for multiple channels by writing into the USB_DIEPx_CTL register followed by a packet write to that endpoint. The core writes the endpoint number, along with the l ast DWORD write for the packet into the Request queue. The core transmits the data in the transmit FIFO when an IN token is received on the USB.

For an OUT transaction, the application sets up a transfer and enables the endpoint. The core receives the OUT data into the receive FIFO, when it has available space. As the packets are received into the FIFO, the application must empty data from it.

From this point on in this chapter, the terms “Pipelined Transaction mode” and “Transfer mode” are used interchangeably.

38.4.2.4 Thresholding in DMA Mode

Note: If the device is operated in threshold mode, the application must ensure that the programmed FIFO sizes are less than or equal to the MaxPacketSize for the corresponding endpoint. If threshold mode is enabled with FIFO size greater than the MaxPacketSize of the endpoint, the device may misbehave. This restriction is applicable for both single and two threshold mode of operation.

The application can program the core to perform FIFO thresholding when operating as a device in DMA mode. With threshold support, the core can be configured to operate with less than maximum packet size FIFOs for a particular endpoint. This results in a smaller FIFO requirement when compared to non-thresholding mode.

FIFO thresholding is supported only in DMA mode. FIFO thresholding is not supported when the core is operating as a host, even in DMA mode.

- The core allows both receive and transmit FIFO thresholding.
- Device Threshold Control Register bit USB_DTHRCTL.RXTHREN must be set to enable receive thresholding. USB_DTHRCTL.RXTHRLEN specifies the receive threshold size.
- Transmit uses separate Threshold Enable controls for isochronous and non-isochronous endpoints. Bits USB_DTHRCTL.NONISOTHREN and USB_DTHRCTL.ISOTHREN specify these Threshold Enable controls.
- The USB_DTHRCTL.TXTHRLEN register field specifies the transmit threshold length the MAC uses to start transmitting data on the bus and is common for isochronous and non-isochronous endpoints. The minimum threshold length the core supports is 8 DWORDs.
- The USB_DTHRCTL.AHBTHRRATIO register field specifies the ratio between AHB TxThreshold and the USB TxThreshold. This ratio allows better use of AHB bandwidth. The following are the programmable values and their resultant AHB TxThreshold values.
  - 0b00: AHB threshold = MAC threshold
  - 0b01: AHB threshold = MAC threshold / 2
  - 0b10: AHB threshold = MAC threshold / 4
  - 0b11: AHB threshold = MAC threshold / 8
- The application must program the threshold value such that the derived AHB threshold does not go below the minimum supported threshold value of four DWORDs as described in 38.4.2.4 Thresholding in DMA Mode.
- Threshold enable controls cannot be changed randomly. The application can set or reset the threshold enable bits only after ensuring that the core is not programmed to do any transfers (FIFOs are flushed, NAK bits are set, all endpoints are disabled).
- One of the limitations of thresholding mode regards possible violation of the violate the PING protocol. A PING token produces an ACK handshake, and the following OUT token could result in a NAK handshake. This behavior is a result of receive FIFO overflow, and cannot be avoided in Thresholding mode. This scenario does not occur when there are no overflows.
- When transmit thresholding is enabled, the core starts transmitting data on the USB for a particular endpoint when there is threshold amount of data available in the corresponding transmit FIFO.
- When receive thresholding is enabled, the core starts transferring data from the receive FIFO to the system memory as soon as there is MAC threshold amount of data available in the receive FIFO. The core handles any underrun or overflow conditions internally.

38.4.3 Host Programming Model

Before you program the Host, read 38.4.1 Overview: Programming the Core and 38.4.2 Modes of Operation.

This section discusses the following topics:

- 38.4.3.1 Channel Initialization
- 38.4.3.2 Halting a Channel
- 38.4.3.3 Sending a Zero-Length Packet in Slave/DMA Modes
- 38.4.3.4 Handling Babble Conditions
- 38.4.3.5 Handling Disconnects
- 38.4.3.6 Host Programming Operations
  - 38.4.3.6.1 Writing the Transmit FIFO in Slave Mode
  - 38.4.3.6.2 Reading the Receive FIFO in Slave Mode

38.4.3.1 Channel Initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps.

1. Program the USB_GINTMSK register to unmask the following:
2. Channel Interrupt
   - Non-periodic Transmit FIFO Empty for OUT transactions (applicable for Slave mode that operates in pipelined transaction-level with the Packet Count field programmed with more than one).
   - Non-periodic Transmit FIFO Half-Empty for OUT transactions (applicable for Slave mode that operates in pipelined transaction-level with the Packet Count field programmed with more than one).
3. Program the USB_USB_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the HCINTMSK register to unmask the transaction-related interrupts of interest given in the Host Channel Interrupt register.
5. Program the selected channel's USB_HCx_TSIZ register.

   Program the register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the selected channels' USB_HCx_DMAADDR register(s) with the buffer start address (DMA mode only).
7. Program the USB_HCx_CHAR register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the Channel Enable bit to 1 only when the application is ready to transmit or receive any packet).

Repeat the above steps for other channels.

Note: De-allocate channel means after the transfer has completed, the channel is disabled. When the application is ready to start the next transfer, the application re-initializes the channel by following these steps.

38.4.3.2 Halting a Channel

The application can disable any channel by programming the USB_HCx_CHAR register with the USB_HCx_CHAR.CHDIS and USB_HCx_CHAR.CHENA bits set to 1. This enables the host to flush the posted requests (if any) and generates a Channel Halted interrupt. The application must wait for the USB_HCx_INT.CHHLTD interrupt before reallocating the channel for other transactions. The host does not interrupt the transaction that has been already started on USB.

In Slave mode operation, before disabling a channel, the application must ensure that there is at least one free space available in the Non-periodic Request Queue (when disabling a non-periodic channel) or the Periodic Request Queue (when disabling a periodic channel). The application can simply flush the posted requests when the Request queue is full (before disabling the channel), by programming the USB_HCx_CHAR register with the USB_HCx_CHAR.CHDIS bit set to 1, and the USB_HCx_CHAR.CHENA bit reset to 0.

The core generates a RXFLVL interrupt when there is an entry in the queue. The application must read/pop the USB_GRXSTSP register to generate the Channel Halted interrupt.

To disable a channel in DMA mode operation, the application need not check for space in the Request queue. The host checks for space in which to write the Disable request on the disabled channel's turn during arbitration. Meanwhile, all posted requests are dropped from the Request queue when the USB_HCx_CHAR.CHDIS bit is set to 1.

The application is expected to disable a channel under any of the following conditions:

1. When a USB_HCx_INT.XFERCOMPL interrupt is received during a non-periodic IN transfer or high-bandwidth interrupt IN transfer (Slave mode only)
2. When a USB_HCx_INT.STALL, USB_HCx_INT.XACTERR, USB_HCx_INT.BBLERR, or USB_HCx_INT.DATATGLERR interrupt is received for an IN or OUT channel (Slave mode only). For high-bandwidth interrupt INs in Slave mode, once the application has received a DATATGLERR interrupt it must disable the channel and wait for a Channel Halted interrupt. The application must be able to receive other interrupts (DATATGLERR, NAK, Data, XACTERR, BBLERR) for the same channel before receiving the halt.
3. When a USB_GINTSTS.DISCONNINT (Disconnect Device) interrupt is received. The application must check for the USB_HPRT.PRTCONNSTS, because when the device directly connected to the host is disconnected, USB_HPRT.PRTCONNSTS is reset. The software must issue a soft reset to ensure that all channels are cleared. When the device is reconnected, the host must issue a USB Reset.
4. When the application aborts a transfer before normal completion (Slave and DMA modes).

Note: In DMA mode, keep the following guideline in mind:

- Channel disable must not be programmed for periodic channels. At the end of the next frame (in the worst case), the core generates a channel halted and disables the channel automatically.

38.4.3.3 Sending a Zero-Length Packet in Slave/DMA Modes

To send a zero-length data packet, the application must initialize an OUT channel as follows.

1. Program the USB_HCx_TSIZ register of the selected channel with a correct PID, XFERSIZE = 0, and PKTCNT = 1.
2. Program the USB_HCx_CHAR register of the selected channel with CHENA = 1 and the device's endpoint characteristics, such as type, speed, and direction.

The application must treat a zero-length data packet as a separate transfer, and cannot combine it with a non-zero-length transfer.

38.4.3.4 Handling Babble Conditions

The core handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

When the core detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already-written data in the Rx buffer and generates a Babble interrupt to the application.

When detects a port babble, it flushes the RxFIFO and disables the port. The core then generates a Port Disabled Interrupt (USB_GINTSTS.PRTINT, USB_HPRT.PRTENCHNG). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the Port Disabled interrupt) by checking USB_HPRT.PRTOVRCURRACT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

38.4.3.5 Handling Disconnects

If the device is disconnected suddenly, a USB_GINTSTS.DISCONNINT interrupt is generated. When the application receives this interrupt, it must issue a soft reset by programming the USB_GRSTCTL.CSFTRST bit.

38.4.3.6 Host Programming Operations

Table 38.1 Host Programming Operations on page 1458 provides links to the programming sequence for the different types of USB transactions.

Table 38.1. Host Programming Operations

Mode	IN	OUT/SETUP
Control		
Slave	38.4.3.6.5 Bulk and Control in Transactions in Slave Mode	38.4.3.6.4 Bulk and Control OUT/SETUP Transactions in Slave Mode
DMA	38.4.3.6.8 Bulk and Control IN Transactions in DMA Mode	38.4.3.6.7 Bulk and Control OUT/SETUP Transactions in DMA Mode
Bulk		
Slave	38.4.3.6.5 Bulk and Control in Transactions in Slave Mode	38.4.3.6.4 Bulk and Control OUT/SETUP Transactions in Slave Mode
DMA	38.4.3.6.8 Bulk and Control IN Transactions in DMA Mode	38.4.3.6.7 Bulk and Control OUT/SETUP Transactions in DMA Mode
Interrupt		
Slave	38.4.3.6.10 Interrupt IN Transactions in Slave Mode	38.4.3.6.9 Interrupt OUT Transactions in Slave Mode
DMA	38.4.3.6.12 Interrupt IN Transactions in DMA Mode	38.4.3.6.11 Interrupt OUT Transactions in DMA Mode
Isochronous		
Slave	38.4.3.6.14 Isochronous IN Transactions in Slave Mode	38.4.3.6.13 Isochronous OUT Transactions in Slave Mode
DMA	38.4.3.6.16 Isochronous IN Transactions in DMA Mode	38.4.3.6.15 Isochronous OUT Transactions in DMA Mode

38.4.3.6.1 Writing the Transmit FIFO in Slave Mode

Figure 38.8 Transmit FIFO Write Task in Slave Mode on page 1459 shows the flow diagram for writing to the transmit FIFO in Slave mode. The host automatically writes an entry (OUT request) to the Periodic/Non-periodic Request Queue, along with the last DWORD write of a packet. The application must ensure that at least one free space is available in the Periodic/Non-periodic Request Queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in DWORDs. If the packet size is non-DWORD aligned, the application must use padding. The host determines the actual packet size based on the programmed maximum packet size and transfer size.

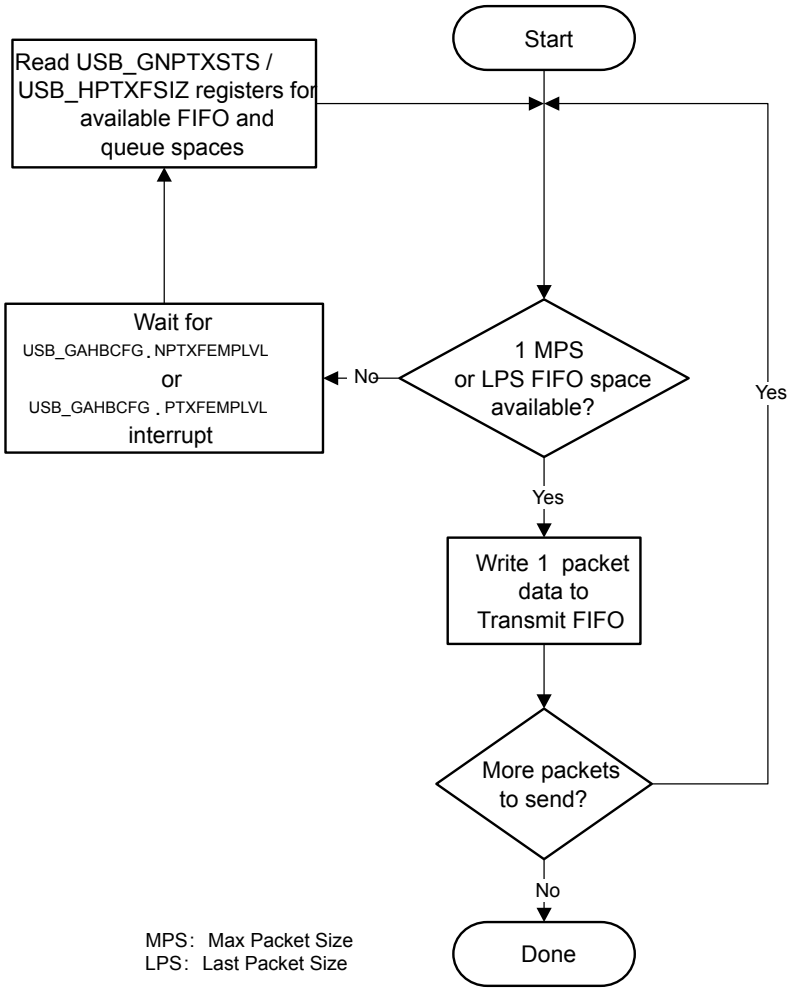


Figure 38.8. Transmit FIFO Write Task in Slave Mode

38.4.3.6.2 Reading the Receive FIFO in Slave Mode

Figure 38.9 Receive FIFO Read Task in Slave Mode on page 1460 shows the flow diagram for reading the receive FIFO in Slave mode. The application must ignore all packet statuses other than IN Data Packet (0b0010).

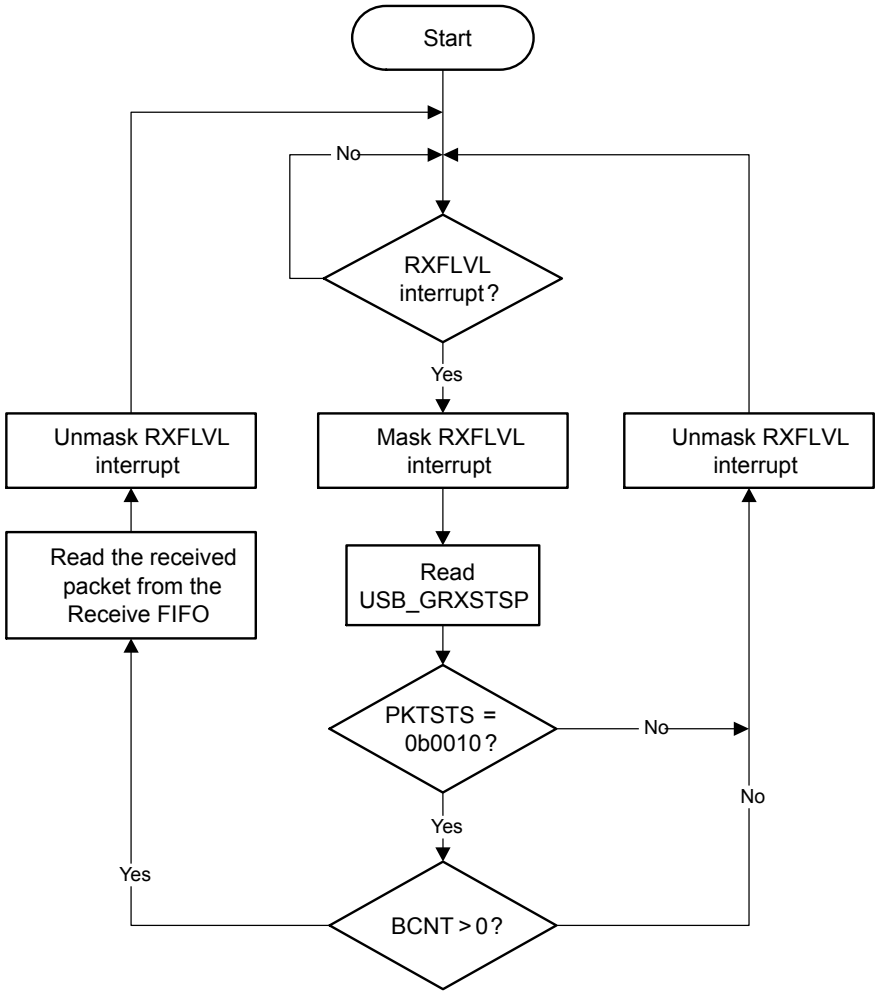


Figure 38.9. Receive FIFO Read Task in Slave Mode

38.4.3.6.3 Control Transactions in Slave Mode

Setup, Data, and Status stages of a control transfer must be performed as three separate transfers. Setup- Data- or Status-stage OUT transactions are performed similarly to the bulk OUT transactions explained in 38.4.3.6.4 Bulk and Control OUT/SETUP Transactions in Slave Mode. Data- or Status-stage IN transactions are performed similarly to the bulk IN transactions explained in 38.4.3.6.5 Bulk and Control in Transactions in Slave Mode For all three stages, the application is expected to set the USB_HC1_CHAR.EPTYPE field to Control. During the Setup stage, the application is expected to set the USB_HC1_TSIZ.PID field to SETUP.

38.4.3.6.4 Bulk and Control OUT/SETUP Transactions in Slave Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization. See Figure 38.8 Transmit FIFO Write Task in Slave Mode on page 1459 and Figure 38.9 Receive FIFO Read Task in Slave Mode on page 1460 for Read or Write data to and from the FIFO in Slave mode.

A typical bulk or control OUT/SETUP pipelined transaction-level operation in Slave mode is shown in Figure 38.10 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode on page 1462. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates the same way but has only one packet. The assumptions are:

- The application is attempting to send two maximum-packet-size packets (transfer size = 1,024 bytes).
- The Non-periodic Transmit FIFO can hold two packets (128 bytes for FS).
- The Non-periodic Request Queue depth = 4.

38.4.3.6.4.1 Normal Bulk and Control OUT/SETUP Operations

The sequence of operations in Figure 38.10 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode on page 1462 (channel 1) is as follows:

1. Initialize channel 1 as explained in 38.4.3.1 Channel Initialization.
2. Write the first packet for channel 1.
3. Along with the last DWORD write, the core writes an entry to the Non-periodic Request Queue.
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame.
5. Write the second (last) packet for channel 1.
6. The core generates the XFERCOMPL interrupt as soon as the last transaction is completed successfully.
7. In response to the XFERCOMPL interrupt, de-allocate the channel for other transfers.

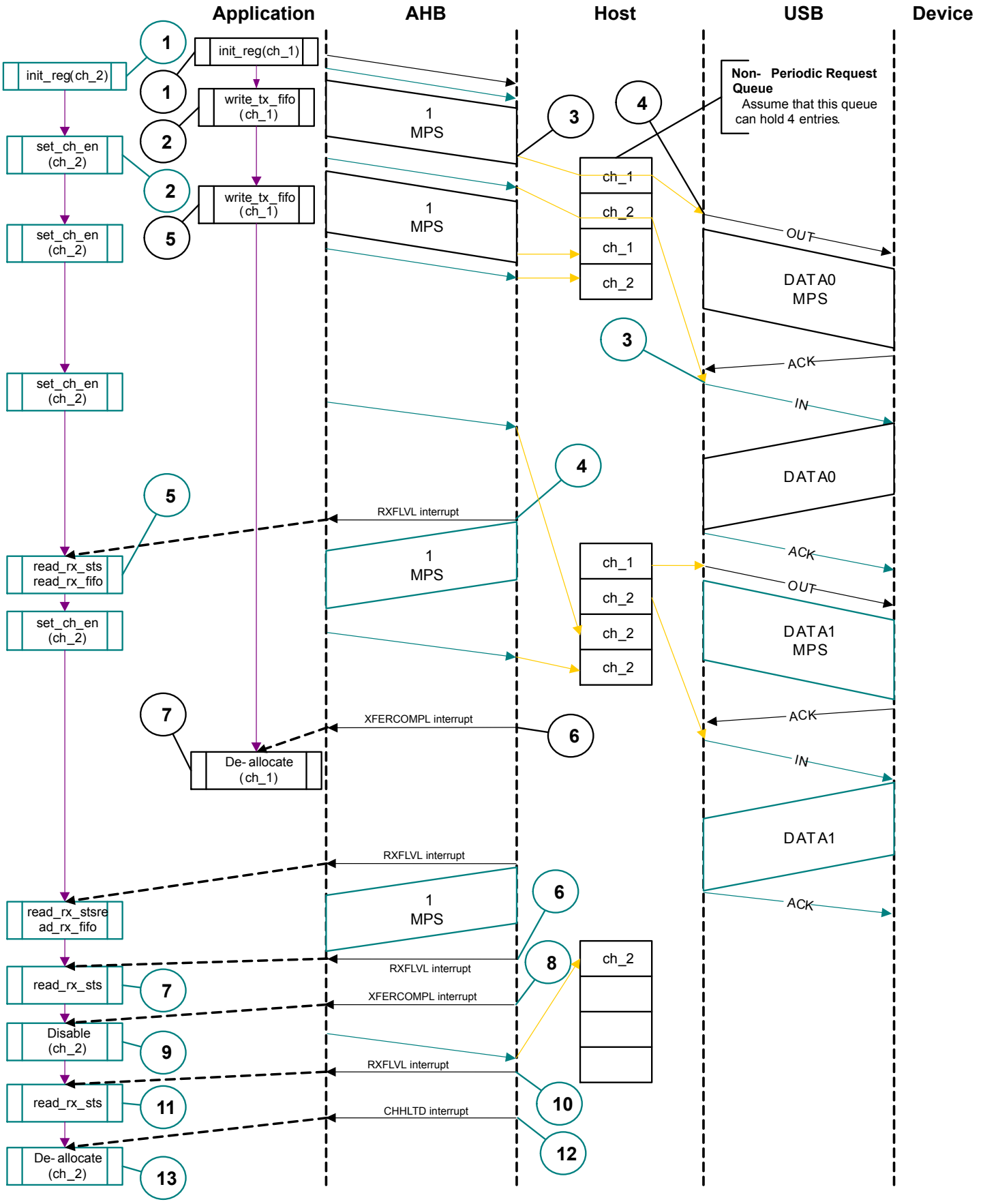


Figure 38.10. Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode

38.4.3.6.4.2 Handling Interrupts

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions in Slave mode is shown in the following code samples.

Interrupt Service Routine for Bulk/Control OUT/SETUP Transactions in Slave Mode

Bulk/Control OUT/SETUP

```
Unmask (NAK/XACTERR/STALL/XFERCOMPL)
if (XFERCOMPL)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else if (STALL)
{
    Transfer Done = 1
    Unmask CHHLTD
    Disable Channel
}
else if (NAK or XACTERR)
{
    Rewind Buffer Pointers
    Unmask CHHLTD
    Disable Channel
    if (XACTERR)
    {
        Increment Error Count
        Unmask ACK
    }
    else
    {
        Reset Error Count
    }
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
```

The application is expected to write the data packets into the transmit FIFO when space is available in the transmit FIFO and the Request queue. The application can make use of USB_GINTSTS.NPTXFEMP interrupt to find the transmit FIFO space.

The application is expected to write the requests as and when the Request queue space is available and until the XFERCOMPL interrupt is received.

38.4.3.6.5 Bulk and Control in Transactions in Slave Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization. See Figure 38.8 Transmit FIFO Write Task in Slave Mode on page 1459 and Figure 38.9 Receive FIFO Read Task in Slave Mode on page 1460 for read or write data to and from the FIFO in Slave mode.

A typical bulk or control IN pipelined transaction-level operation in Slave mode is shown in Figure 38.10 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode on page 1462. See channel 2 (ch_2). The assumptions are:

1. The application is attempting to receive two maximum-sized packets (transfer size = 1,024 bytes).
2. The receive FIFO can contain at least one maximum-packet-size packet and two status DWORDs per packet (72 bytes for FS).
3. The Non-periodic Request Queue depth = 4.

38.4.3.6.5.1 Normal Bulk and Control IN Operations

The sequence of operations in Figure 38.10 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode on page 1462 is as follows:

1. Initialize channel 2 as explained in 38.4.3.1 Channel Initialization.
2. Set the USB_HC2_CHAR.CHENA bit to write an IN request to the Non-periodic Request Queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (USB_GRXSTSR.PKTSTS != 0b0010).
8. The core generates the XFERCOMPL interrupt as soon as the receive packet status is read.
9. In response to the XFERCOMPL interrupt, disable the channel (see 38.4.3.2 Halting a Channel) and stop writing the USB_HC2_CHAR register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the USB_HC2_CHAR register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHHLTD interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHHLTD interrupt, de-allocate the channel for other transfers.

Note: For Bulk/Control IN transfers, the application must write the requests when the Request queue space is available, and until the XFERCOMPL interrupt is received.

38.4.3.6.5.2 Handling Interrupts

The channel-specific interrupt service routine for bulk and control IN transactions in Slave mode is shown in the following code samples.

Interrupt Service Routine for Bulk/Control IN Transactions in Slave Mode

```
Unmask (XACTERR/XFERCOMPL/BBLERR/STALL/DATATGLERR)
if (XFERCOMPL)
{
    Reset Error Count
    Unmask CHHLTD
    Disable Channel
    Reset Error Count
    Mask ACK
}
else if (XACTERR or BBLERR or STALL)
{
    Unmask CHHLTD
    Disable Channel
    if (XACTERR)
    {
        Increment Error Count
        Unmask ACK
    }
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
else if (DATATGLERR)
{
    Reset Error Count
}
```

38.4.3.6.6 Control Transactions in DMA Mode

Setup, Data, and Status stages of a control transfer must be performed as three separate transfers. Setup- and Data- or Status-stage OUT transactions are performed similarly to the bulk OUT transactions explained in 38.4.3.6.7 Bulk and Control OUT/SETUP Transactions in DMA Mode. Data- or Status-stage IN transactions are performed similarly to the bulk IN transactions explained in 38.4.3.6.8 Bulk and Control IN Transactions in DMA Mode. For all three stages, the application is expected to set the USB_HC1_CHAR.EPTYPE field to Control. During the Setup stage, the application is expected to set the USB_HC1_TSIZ.PID field to SETUP.

38.4.3.6.7 Bulk and Control OUT/SETUP Transactions in DMA Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization.

This section discusses the following topics:

- 38.4.3.6.7.1 Overview
- 38.4.3.6.7.2 Normal Bulk and Control OUT/SETUP Operations
- 38.4.3.6.7.3 NAK Handling With DMA
- 38.4.3.6.7.4 Handling Interrupts

38.4.3.6.7.1 Overview

- The application is attempting to send two maximum-packet-size packets (transfer size = 1,024 bytes).
- The Non-periodic Transmit FIFO can hold two packets (128 bytes for FS).
- The Non-periodic Request Queue depth = 4.

38.4.3.6.7.2 Normal Bulk and Control OUT/SETUP Operations

The sequence of operations in Figure 38.10 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode on page 1462 is as follows:

1. Initialize and enable channel 1 as explained in 38.4.3.1 Channel Initialization.
2. The host starts fetching the first packet as soon as the channel is enabled. For DMA mode, the host uses the programmed DMA address to fetch the packet.
3. After fetching the last DWORD of the second (last) packet, the host masks channel 1 internally for further arbitration.
4. The host generates a CHHLTD interrupt as soon as the last packet is sent.
5. In response to the CHHLTD interrupt, de-allocate the channel for other transfers.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions in DMA mode is shown in 38.4.3.6.7.4 Handling Interrupts.

38.4.3.6.7.3 NAK Handling With DMA

1. The Host sends a Bulk OUT Transaction.
2. The Device responds with NAK.
3. If the application has unmasked NAK, the core generates the corresponding interrupt(s) to the application.

   The application is not required to service these interrupts, since the core takes care of rewinding of buffer pointers and re-initializing the Channel without application intervention.
4. When the Device returns an ACK, the core continues with the transfer.

Optionally, the application can utilize these interrupts. If utilized by the application:

- The NAK interrupt is masked by the application.
- The core does not generate a separate interrupt when NAK is received by the Host functionality.

Application Programming Flow

1. The application programs a channel to do a bulk transfer for a particular data size in each transaction.
   - Packet Data size can be up to 512 KBytes
   - Zero-length data must be programmed as a separate transaction.
2. Program the transfer size register with:
   - Transfer size
   - Packet Count
3. Program the DMA address.
4. Program the USB_HCx_CHAR to enable the channel.
5. The Interrupt handling by the application is as depicted in the flow diagram.

Note: The NAK interrupts are still generated internally. The application can mask off these interrupts from reaching it. The application can use these interrupts optionally.

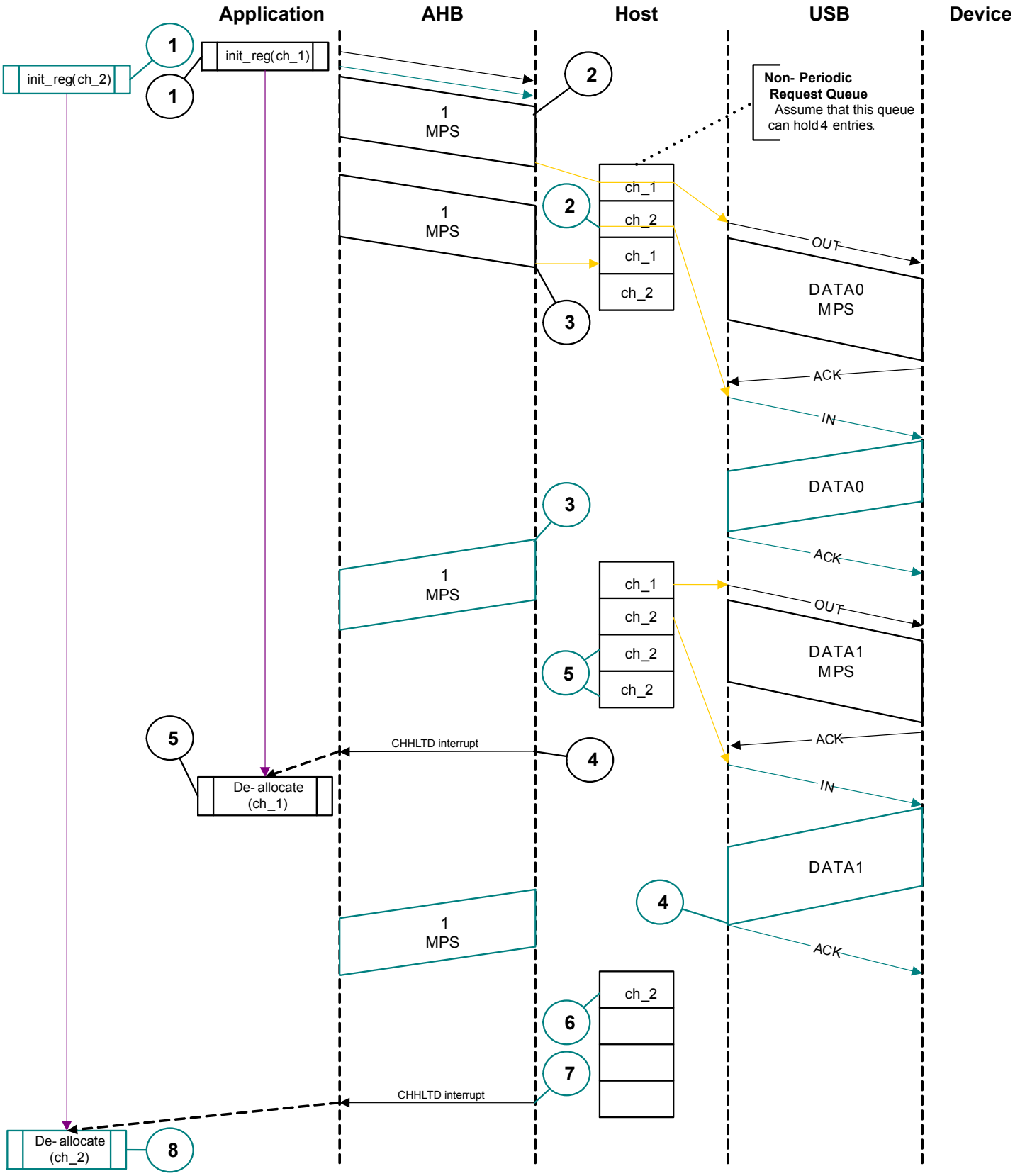


Figure 38.11. Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in DMA Mode

38.4.3.6.7.4 Handling Interrupts

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions in DMA mode is shown in the following code samples.

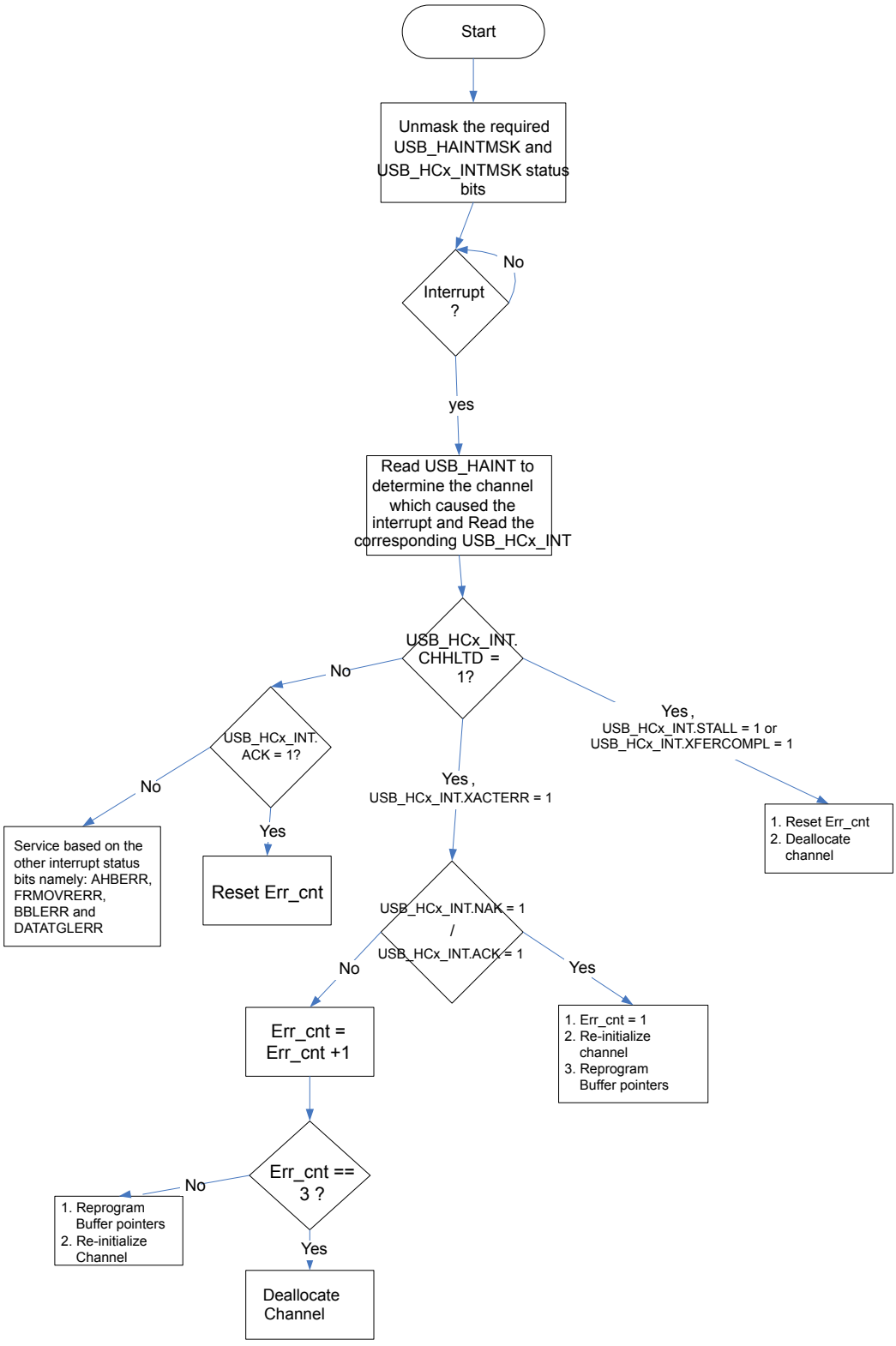


Figure 38.12. Interrupt Service Routine for Bulk/Control OUT Transaction in DMA Mode

In Figure 38.12 Interrupt Service Routine for Bulk/Control OUT Transaction in DMA Mode on page 1469 that the Interrupt Service Routine is not required to handle NAK responses. This is the difference of proposed flow with respect to current flow. Similar flow is applicable for Control flow also.

The NAK status bits in USB_HCx_INT registers are updated. The application can unmask these interrupts when it requires the core to generate an interrupt for NAK. The NAK status is updated because during Xact_err scenarios, this status provides a means for the application to determine whether the Xact_err occurred three times consecutively or there were NAK responses in between two Xact_err. This provides a mechanism for the application to reset the error counter accordingly. The application must read the NAK/ACK along with the xact_err. If NAK/ACK is not set, the Xact_err count must be incremented otherwise application must initialize the Xact_err count to 1.

Bulk/Control OUT/SETUP

```
Unmask (CHHLTD)
if (CHHLTD)
{
    if (XFERCOMPL or STALL)
    {
        Reset Error Count (Error_count=1)
        Mask ACK
        De-allocate Channel
    }
    else if (XACTERR)
    {
        if (NAK/ACK)
        {
            Error_count = 1
            Re-initialize Channel
            Rewind Buffer Pointers
        }
        else
        {
            Error_count = Error_count + 1
            if (Error_count == 3)
            {
                De allocate channel
            }
            else
            {
                Re-initialize Channel
                Rewind Buffer Pointers
            }
        }
    }
}
else if (ACK)
{
    Reset Error Count (Error_count=1)
    Mask ACK
}
```

As soon as the channel is enabled, the core attempts to fetch and write data packets, in multiples of the maximum packet size, to the transmit FIFO when space is available in the transmit FIFO and the Request queue. The core stops fetching as soon as the last packet is fetched.

38.4.3.6.8 Bulk and Control IN Transactions in DMA Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization.

A typical bulk or control IN operation in DMA mode is shown in Figure 38.11 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in DMA Mode on page 1468. See channel 2 (ch_2).

The assumptions are:

1. The application is attempting to receive two maximum-packet-size packets (transfer size = 1,024 bytes).
2. The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (72 bytes for FS).
3. The Non-periodic Request Queue depth = 4.

38.4.3.6.8.1 Normal Bulk and Control IN Operations

The sequence of operations in Figure 38.11 Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in DMA Mode on page 1468 is as follows:

1. Initialize and enable channel 2 as explained in 38.4.3.1 Channel Initialization.
2. The host writes an IN request to the Request queue as soon as channel 2 receives the grant from the arbiter. (Arbitration is performed in a round-robin fashion, with fairness.).
3. The host starts writing the received data to the system memory as soon as the last byte is received with no errors.
4. When the last packet is received, the host sets an internal flag to remove any extra IN requests from the Request queue.
5. The host flushes the extra requests.
6. The final request to disable channel 2 is written to the Request queue. At this point, channel 2 is internally masked for further arbitration.
7. The host generates the CHHLTD interrupt as soon as the disable request comes to the top of the queue.
8. In response to the CHHLTD interrupt, de-allocate the channel for other transfers.

38.4.3.6.8.2 Handling Interrupts

The channel-specific interrupt service routine for bulk and control IN transactions in DMA mode is shown in the following flow:

Interrupt Service Routines for Bulk/Control Bulk/Control IN Transactions in DMA Mode

Bulk/Control IN

```
Unmask (CHHLTD)
if (CHHLTD)
{
    if (XFERCOMPL or STALL or BBLERR)
    {
        Reset Error Count Mask ACK De-allocate Channel
    }
    else if (XACTERR)
    {
        if (Error_count == 2)
        {
            De-allocate Channel
        }
        else
        {
            Unmask ACK
            Unmask NAK
            Unmask DATATGLERR
            Increment Error
            Count Re-initialize Channel
        }
    }
}
else if (ACK or NAK or DATATGLERR)
{
    Reset Error Count
    Mask ACK
    Mask NAK
    Mask DATATGLERR
}
```

38.4.3.6.9 Interrupt OUT Transactions in Slave Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization. See Figure 38.8 Transmit FIFO Write Task in Slave Mode on page 1459 and Figure 38.9 Receive FIFO Read Task in Slave Mode on page 1460 for read or write data to and from the FIFO in Slave mode.

A typical interrupt OUT operation in Slave mode is shown in Figure 38.13 Normal Interrupt OUT/IN Transactions in Slave Mode on page 1473. See channel 1 (ch_1). The assumptions are:

- The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1,024 bytes).
- The Periodic Transmit FIFO can hold one packet.
- Periodic Request Queue depth = 4.

38.4.3.6.9.1 Normal Interrupt OUT Operation

The sequence of operations in Figure 38.13 Normal Interrupt OUT/IN Transactions in Slave Mode on page 1473 is as follows:

1. Initialize and enable channel 1 as explained in 38.4.3.1 Channel Initialization. The application must set the USB_HC1_CHAR.ODDFRM bit.
2. Write the first packet for channel 1. For a high-bandwidth interrupt transfer, the application must write the subsequent packets up to MC (maximum number of packets to be transmitted in the next frame times before switching to another channel).
3. Along with the last DWORD write of each packet, the host writes an entry to the Periodic Request Queue.
4. The host attempts to send an OUT token in the next (odd) frame.
5. The host generates an XFERCOMPL interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFERCOMPL interrupt, reinitialize the channel for the next transfer.

38.4.3.6.9.2 Handling Interrupts

The channel-specific interrupt service routine for Interrupt OUT transactions in Slave mode is shown in the following flow:

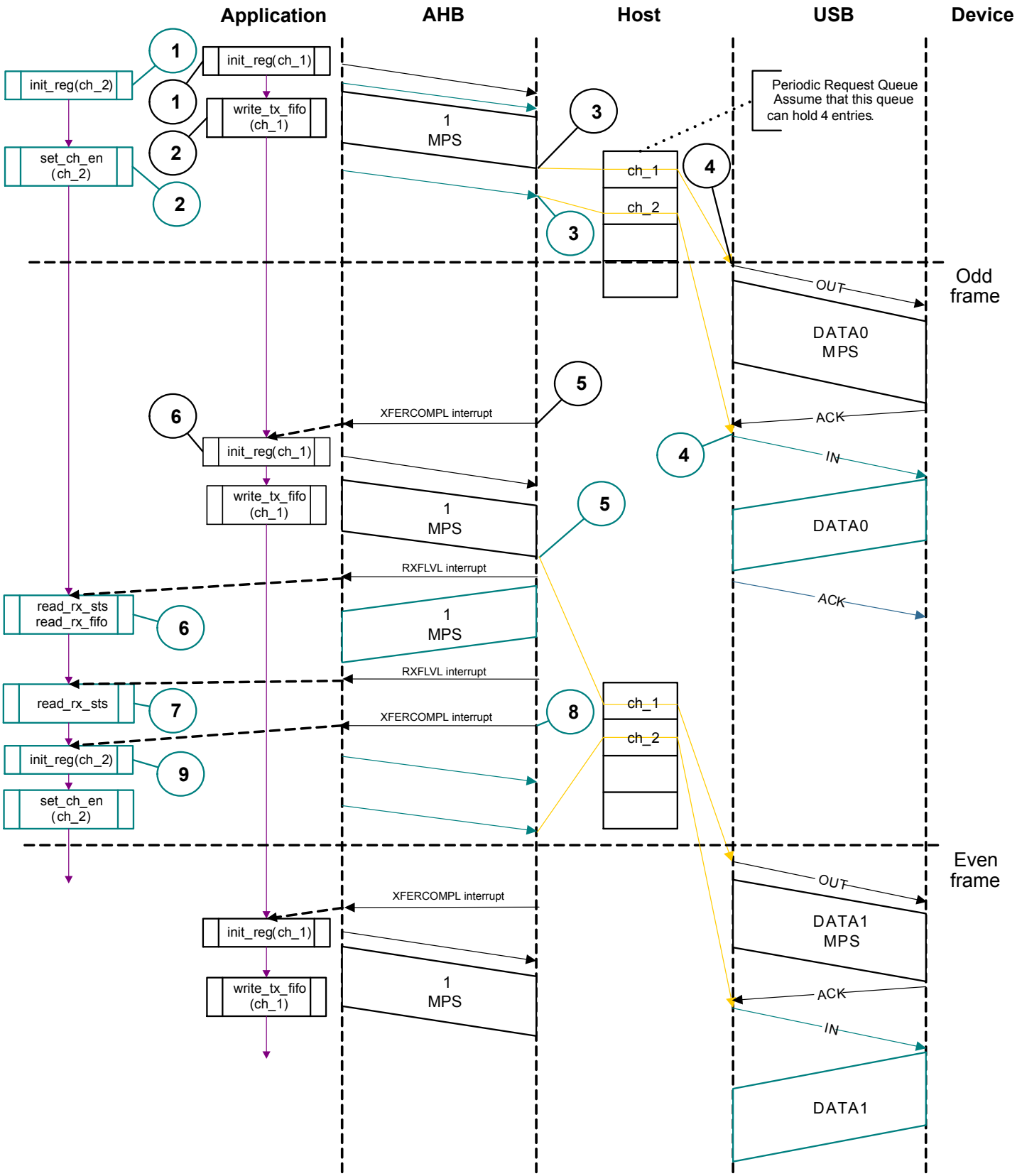


Figure 38.13. Normal Interrupt OUT/IN Transactions in Slave Mode

Interrupt Service Routine for Interrupt OUT Transactions in Slave Mode

Interrupt OUT

```
Unmask (NAK/XACTERR/STALL/XFERCOMPL/FRMOVRUN)
if (XFERCOMPL)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else if (STALL or FRMOVRUN)
{
    Mask ACK
    Unmask CHHLTD
    Disable Channel
    if (STALL)
    {
        Transfer Done = 1
    }
}
else if (NAK or XACTERR)
{
    Rewind Buffer Pointers
    Reset Error Count
    Mask ACK
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel (in next b_interval - 1 Frame)
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the Request queue up to the count specified in the MC field before switching to another channel. The application uses the USB_GINTSTS.NPTXFEMP interrupt to find the transmit FIFO space.

38.4.3.6.10 Interrupt IN Transactions in Slave Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization. See Transmit FIFO Write Task in Slave Mode and Receive FIFO Read Task in Slave Mode for read or write data to and from the FIFO in Slave mode.

A typical interrupt-IN operation in Slave mode is shown in Figure 38.13 Normal Interrupt OUT/IN Transactions in Slave Mode on page 1473. See channel 2 (ch_2). The assumptions are:

1. The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd. (transfer size = 1,024 bytes).
2. The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (1,031 bytes for FS).
3. Periodic Request Queue depth = 4.

38.4.3.6.10.1 Normal Interrupt IN Operation

The sequence of operations in Figure 38.13 Normal Interrupt OUT/IN Transactions in Slave Mode on page 1473 (channel 2) is as follows:

1. Initialize channel 2 as explained in 38.4.3.1 Channel Initialization. The application must set the USB_HC2_CHAR.ODDFRM bit.
2. Set the USB_HC2_CHAR.CHENA bit to write an IN request to the Periodic Request Queue. For a high-bandwidth interrupt transfer, the application must write the USB_HC2_CHAR register MC (maximum number of expected packets in the next frame) times before switching to another channel.
3. The host writes an IN request to the Periodic Request Queue for each USB_HC2_CHAR register write with a CHENA bit set.
4. The host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (USB_GRXSTSR.PKTSTS != 0b0010).
8. The core generates an XFERCOMPL interrupt as soon as the receive packet status is read.
9. In response to the XFERCOMPL interrupt, read the USB_HC2_TSIZ.PKTCNT field. If USB_HC2_TSIZ.PKTCNT != 0, disable the channel (as explained in 38.4.3.2 Halting a Channel) before re-initializing the channel for the next transfer, if any). If USB_HC2_TSIZ.PKTCNT == 0, reinitialize the channel for the next transfer. This time, the application must reset the USB_HC2_CHAR.ODDFRM bit.

38.4.3.6.10.2 Handling Interrupts

The channel-specific interrupt service routine for an interrupt IN transaction in Slave mode is a follows.

Interrupt IN

```
Unmask (NAK/XACTERR/XFERCOMPL/BBLERR/STALL/FRMOVRUN/DATATGLERR)
if (XFERCOMPL)
{
    Reset Error Count
    Mask ACK
    if (USB_HCx_TSIZ.PKTCNT == 0)
    {
        De-allocate Channel
    }
    else
    {
        Transfer Done = 1
        Unmask CHHLTD
        Disable Channel
    }
}
else if (STALL or FRMOVRUN or NAK or DATATGLERR or BBLERR)
{
    Mask ACK
    Unmask CHHLTD
    Disable Channel
    if (STALL or BBLERR)
    {
        Reset Error Count
        Transfer Done = 1
    }
    else if (!FRMOVRUN)
    {
        Reset Error Count
    }
}
else if (XACTERR)
{
    Increment Error Count
    Unmask ACK
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel (in next b_interval - 1 Frame)
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
```

The application is expected to write the requests for the same channel when the Request queue space is available up to the count specified in the MC field before switching to another channel (if any).

38.4.3.6.11 Interrupt OUT Transactions in DMA Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization.

A typical interrupt OUT operation in DMA mode is shown in Figure 38.14 Normal Interrupt OUT/IN Transactions in DMA Mode on page 1479. See channel 1 (ch_1). The assumptions are:

- The application is attempting to transmit one packet in every frame (up to 1 maximum packet size of 1,024 bytes).
- The Periodic Transmit FIFO can hold one packet (1 KB for FS).
- Periodic Request Queue depth = 4.

38.4.3.6.11.1 Normal Interrupt OUT Operation

1. Initialize and enable channel 1 as explained in 38.4.3.1 Channel Initialization.
2. The host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last DWORD fetch. In high-bandwidth transfers, the host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The host attempts to send the OUT token in the beginning of the next odd frame.
4. After successfully transmitting the packet, the host generates a CHHLTD interrupt.
5. In response to the CHHLTD interrupt, reinitialize the channel for the next transfer.

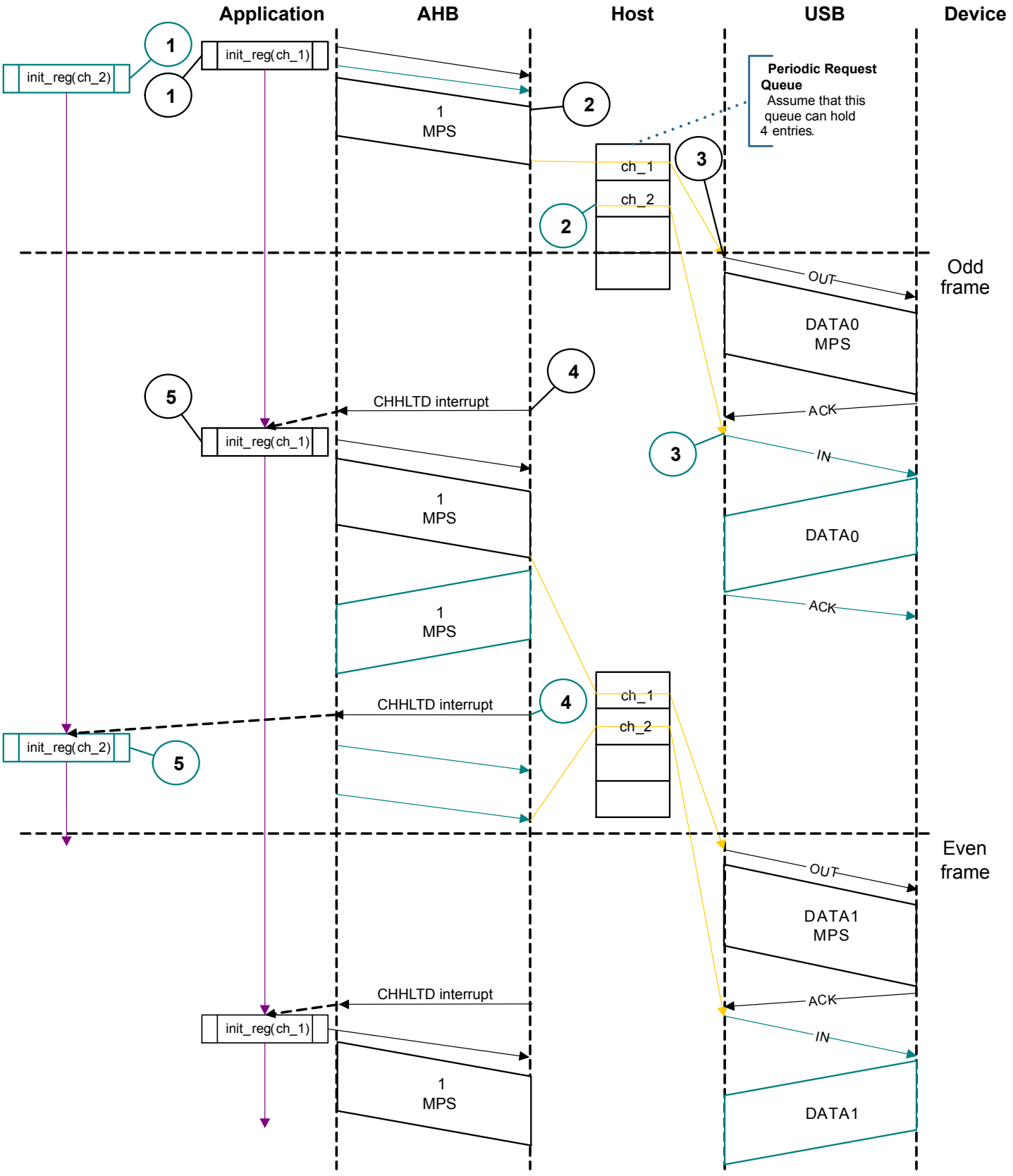


Figure 38.14. Normal Interrupt OUT/IN Transactions in DMA Mode

38.4.3.6.11.2 Handling Interrupts

The following code sample shows the channel-specific ISR for an interrupt OUT transaction in DMA mode.

Interrupt OUT

```
Unmask (CHHLTD)
if (CHHLTD)
{
    if (XFERCOMPL)
    {
        Reset Error Count
        Mask ACK
        if (Transfer Done)
        {
            De-allocate Channel
        }
        else
        {
            Re-initialize Channel (in next b_interval - 1 Frame)
        }
    }
    else if (STALL)
    {
        Transfer Done = 1
        Reset Error Count
        Mask ACK
        De-allocate Channel
    }
    else if (NAK or FRMOVRUN)
    {
        Mask ACK
        Rewind Buffer Pointers
        Re-initialize Channel (in next b_interval - 1 Frame)
        if (NAK)
        {
            Reset Error Count
        }
    }
    else if (XACTERR)
    {
        if (Error_count == 2)
        {
            De-allocate Channel
        }
        else
        {
            Increment Error Count
            Rewind Buffer Pointers
            Unmask ACK
            Re-initialize Channel (in next b_interval - 1 Frame)
        }
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
```

As soon as the channel is enabled, the core attempts to fetch and write data packets, in maximum packet size multiples, to the transmit FIFO when the space is available in the transmit FIFO and the Request queue. The core stops fetching as soon as the last packet is fetched (the number of packets is determined by the MC field of the USB_HCx_CHAR register).

38.4.3.6.12 Interrupt IN Transactions in DMA Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization.

A typical interrupt IN operation in DMA mode is shown in Figure 38.14 Normal Interrupt OUT/IN Transactions in DMA Mode on page 1479. See channel 2 (ch_2). The assumptions are:

- The application is attempting to receive one packet in every frame (up to 1 maximum packet size of 1,024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (1,032 bytes for FS).
- Periodic Request Queue depth = 4.

38.4.3.6.12.1 Normal Interrupt IN Operation

The sequence of operations in Figure 38.14 Normal Interrupt OUT/IN Transactions in DMA Mode on page 1479 (channel 2) is as follows:

1. Initialize and enable channel 2 as explained in 38.4.3.1 Channel Initialization.
2. The host writes an IN request to the Request queue as soon as the channel 2 gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the host writes consecutive writes up to MC times.
3. The host attempts to send an IN token at the beginning of the next (odd) frame.
4. As soon the packet is received and written to the receive FIFO, the host generates a CHHLTD interrupt.
5. In response to the CHHLTD interrupt, reinitialize the channel for the next transfer.

38.4.3.6.12.2 Handling Interrupts

The channel-specific interrupt service routine for Interrupt IN transactions in DMA mode is as follows.

Interrupt Service Routine for Interrupt IN Transactions in DMA Mode

```
Unmask (CHHLTD)
if (CHHLTD)
{
    if (XFERCOMPL)
    {
        Reset Error Count
        Mask ACK
        if (Transfer Done)
        {
            De-allocate Channel
        }
        else
        {
            Re-initialize Channel (in next b_interval - 1 Frame)
        }
    }
    else if (STALL or BBLERR)
    {
        Reset Error Count
        Mask ACK
        De-allocate Channel
    }
    else if (NAK or DATATGLERR or FRMOVRUN)
    {
        Mask ACK
        Re-initialize Channel (in next b_interval - 1 Frame)
        if (DATATGLERR or NAK)
        {
            Reset Error Count
        }
    }
    else if (XACTERR)
    {
        if (Error_count == 2)
        {
            De-allocate Channel
        }
        else
        {
            Increment Error Count
            Unmask ACK
            Re-initialize Channel (in next b_interval - 1 Frame)
        }
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
```

As soon as the channel is enabled, the core attempts to write the requests into the Request queue when the space is available up to the count specified in the MC field.

38.4.3.6.13 Isochronous OUT Transactions in Slave Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization. See TFigure 38.8 Transmit FIFO Write Task in Slave Mode on page 1459 and Figure 38.9 Receive FIFO Read Task in Slave Mode on page 1460 for read or write data to and from the FIFO in Slave mode.

A typical isochronous OUT operation in Slave mode is shown in Figure 38.15 Normal Isochronous OUT/IN Transactions in Slave Mode on page 1484. See channel 1 (ch_1). The assumptions are:

- The application is attempting to send one packet every frame (up to 1 maximum packet size), starting with an odd frame. (transfer size = 1,024 bytes).
- The Periodic Transmit FIFO can hold one packet (1 KB).
- Periodic Request Queue depth = 4.

38.4.3.6.13.1 Normal Isochronous OUT Operation

The sequence of operations in Figure 38.15 Normal Isochronous OUT/IN Transactions in Slave Mode on page 1484 (channel 1) is as follows:

1. Initialize and enable channel 1 as explained in 38.4.3.1 Channel Initialization. The application must set the USB_HC1_CHAR.ODDFRM bit.
2. Write the first packet for channel 1. For a high-bandwidth isochronous transfer, the application must write the subsequent packets up to MC (maximum number of packets to be transmitted in the next frame) times before switching to another channel.
3. Along with the last DWORD write of each packet, the host writes an entry to the Periodic Request Queue.
4. The host attempts to send the OUT token in the next frame (odd).
5. The host generates the XFERCOMPL interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFERCOMPL interrupt, reinitialize the channel for the next transfer.

38.4.3.6.13.2 Handling Interrupts

The channel-specific interrupt service routine for isochronous OUT transactions in Slave mode is shown in the following flow:

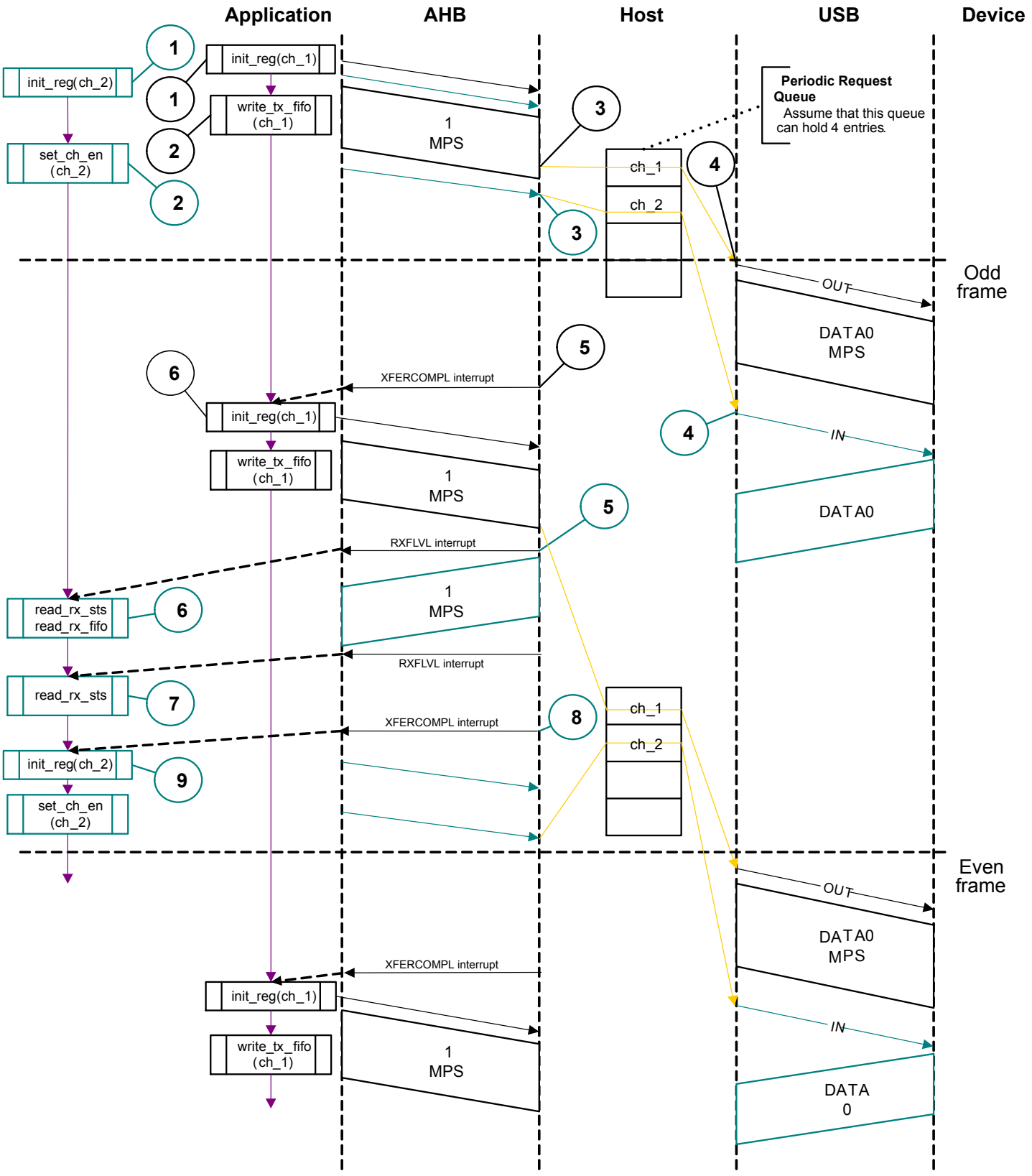


Figure 38.15. Normal Isochronous OUT/IN Transactions in Slave Mode

Interrupt Service Routine for Isochronous OUT Transactions in Slave Mode

Isochronous OUT

```
Unmask (FRMOVRUN/XFERCOMPL)
if (XFERCOMPL)
{
    De-allocate Channel
}
else if (FRMOVRUN)
{
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    De-allocate Channel
}
```

38.4.3.6.14 Isochronous IN Transactions in Slave Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization. See Figure 38.8 Transmit FIFO Write Task in Slave Mode on page 1459 and Figure 38.9 Receive FIFO Read Task in Slave Mode on page 1460 for read or write data to and from the FIFO in Slave mode.

A typical isochronous IN operation in Slave mode is shown in Figure 38.15 Normal Isochronous OUT/IN Transactions in Slave Mode on page 1484. See channel 2 (ch_2). The assumptions are:

- The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame. (transfer size = 1,024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (1,031 bytes for FS).
- Periodic Request Queue depth = 4.

38.4.3.6.14.1 Normal Isochronous IN Operation

The sequence of operations in Figure 38.15 Normal Isochronous OUT/IN Transactions in Slave Mode on page 1484 (channel 2) is as follows:

1. Initialize channel 2 as explained in 38.4.3.1 Channel Initialization. The application must set the USB_HC2_CHAR.ODDFRM bit.
2. Set the USB_HC2_CHAR.CHENA bit to write an IN request to the Periodic Request Queue. For a high-bandwidth isochronous transfer, the application must write the USB_HC2_CHAR register MC (maximum number of expected packets in the next frame) times before switching to another channel.
3. The host writes an IN request to the Periodic Request Queue for each USB_HC2_CHAR register write with the CHENA bit set.
4. The host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (USB_GRXSTSR.PKTSTS != 0b0010).
8. The core generates an XFERCOMPL interrupt as soon as the receive packet status is read.
9. In response to the XFERCOMPL interrupt, read the USB_HC2_TSIZ.PKTCNT field. If USB_HC2_TSIZ.PKTCNT != 0, disable the channel (as explained in 38.4.3.2 Halting a Channel) before re-initializing the channel for the next transfer, if any. If USB_HC2_TSIZ.PKTCNT == 0, reinitialize the channel for the next transfer. This time, the application must reset the USB_HC2_CHAR.ODDFRM bit.

38.4.3.6.14.2 Handling Interrupts

The channel-specific interrupt service routine for an isochronous IN transaction in Slave mode is as follows.

Isochronous IN

```
Unmask (XACTERR/XFERCOMPL/FRMOVRUN/BBLERR)
if (XFERCOMPL or FRMOVRUN)
{
    if (XFERCOMPL and (USB_HCx_TSIZ.PKTCNT == 0))
    {
        Reset Error Count
        De-allocate Channel
    }
    else
    {
        Unmask CHHLTD
        Disable Channel
    }
}
else if (XACTERR or BBLERR)
{
    Increment Error Count
    Unmask CHHLTD
    Disable Channel
}
else if (CHHLTD)
{
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
```

38.4.3.6.15 Isochronous OUT Transactions in DMA Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization.

A typical isochronous OUT operation in DMA mode is shown in Figure 38.16 Normal Isochronous OUT/IN Transactions in DMA Mode on page 1487. See channel 1 (ch_1). The assumptions are:

- The application is attempting to transmit one packet every frame (up to 1 maximum packet size of 1,024 bytes).
- The Periodic Transmit FIFO can hold one packet (1 KB).
- Periodic Request Queue depth = 4.

38.4.3.6.15.1 Normal Isochronous OUT Operation

1. Initialize and enable channel 1 as explained in 38.4.3.1 Channel Initialization.
2. The host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last DWORD fetch. In high-bandwidth transfers, the host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The host attempts to send an OUT token in the beginning of the next (odd) frame.
4. After successfully transmitting the packet, the host generates a CHHLTD interrupt.
5. In response to the CHHLTD interrupt, reinitialize the channel for the next transfer.

38.4.3.6.15.2 Handling Interrupts

The channel-specific interrupt service routine for Isochronous OUT transactions in DMA mode is shown in the following flow:

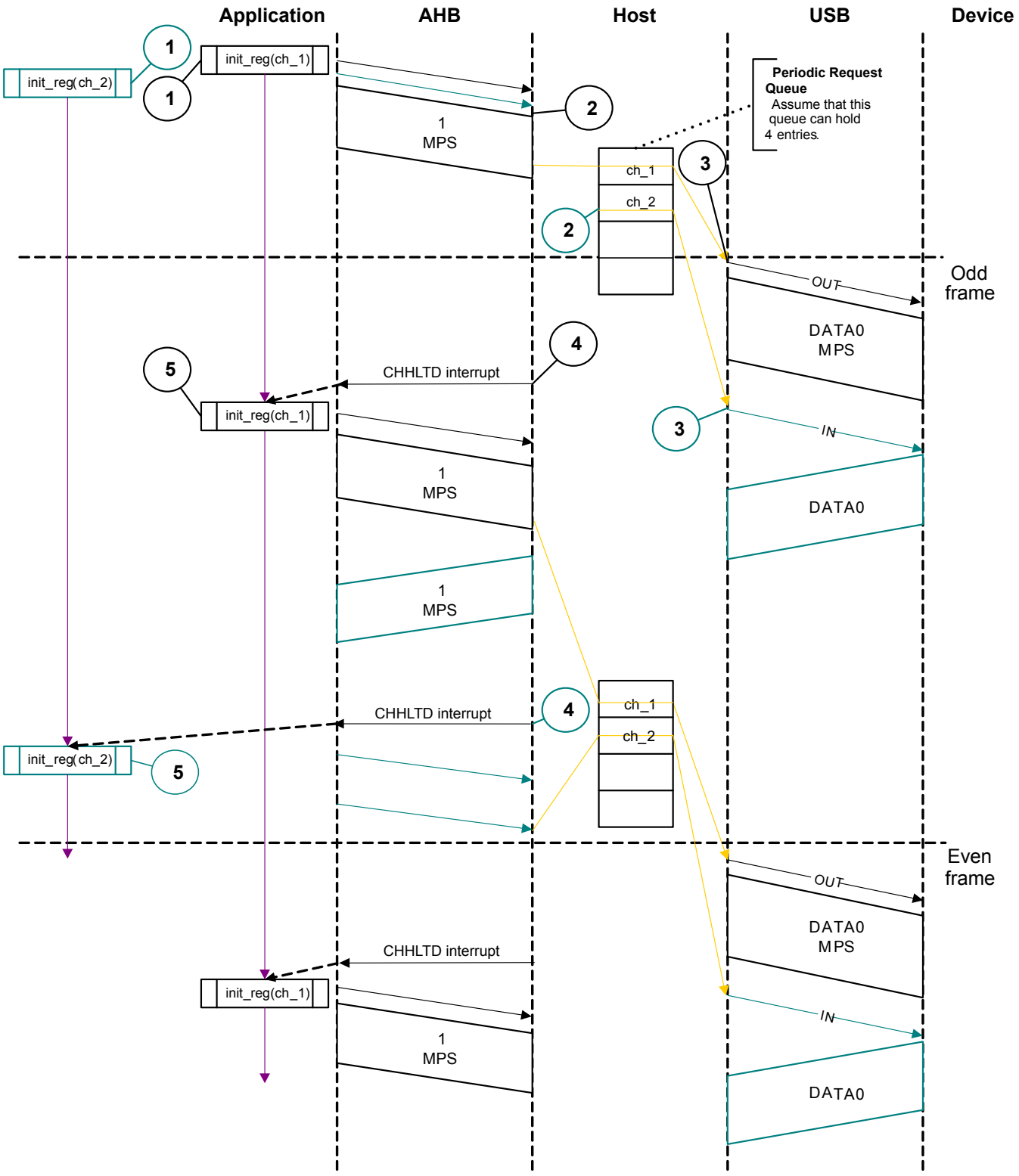


Figure 38.16. Normal Isochronous OUT/IN Transactions in DMA Mode

Interrupt Service Routine for Isochronous OUT Transactions in DMA Mode

Isochronous OUT

```
Unmask (CHHLTD)
if (CHHLTD)
{
    if (XFERCOMPL or FRMOVRUN)
    {
        De-allocate Channel
    }
}
```

38.4.3.6.16 Isochronous IN Transactions in DMA Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 38.4.3.1 Channel Initialization.

A typical isochronous IN operation in DMA mode is shown in Figure 38.16 Normal Isochronous OUT/IN Transactions in DMA Mode on page 1487. See channel 2 (ch_2). The assumptions are:

- The application is attempting to receive one packet in every frame (up to 1 maximum packet size of 1,024 bytes).
- The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDS per packet (1,031 bytes).
- Periodic Request Queue depth = 4.

38.4.3.6.16.1 Normal Isochronous IN Operation

The sequence of operations in Figure 38.16 Normal Isochronous OUT/IN Transactions in DMA Mode on page 1487 (channel 2) is as follows:

1. Initialize and enable channel 2 as explained in 38.4.3.1 Channel Initialization.
2. The host writes an IN request to the Request queue as soon as the channel 2 gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the host performs consecutive writes up to MC times.
3. The host attempts to send an IN token at the beginning of the next (odd) frame.
4. As soon the packet is received and written to the receive FIFO, the host generates a CHHLTD interrupt.
5. In response to the CHHLTD interrupt, reinitialize the channel for the next transfer.

38.4.3.6.16.2 Handling Interrupts

The channel-specific interrupt service routine for an isochronous IN transaction in DMA mode is as follows.

Isochronous IN

```
Unmask (CHHLTD)
if (CHHLTD)
{
    if (XFERCOMPL or FRMOVRUN)
    {
        if (XFERCOMPL and (USB_HCx_TSIZ.PKTCNT == 0))
        {
            Reset Error Count
            De-allocate Channel
        }
        else
        {
            De-allocate Channel
        }
    }
    else if (XACTERR or BBLERR)
    {
        if (Error_count == 2)
        {
            De-allocate Channel
        }
        else
        {
            Increment Error Count
            Re-enable Channel (in next b_interval - 1 Frame)
        }
    }
}
```

38.4.4 Device Programming Model

Before you program the Device, be sure to read 38.4.1 Overview: Programming the Core and 38.4.2 Modes of Operation

38.4.4.1 Endpoint Initialization

This section addresses the following topics:

- 38.4.4.1.1 Initialization on USB Reset
- 38.4.4.1.2 Initialization on Enumeration Completion
- 38.4.4.1.3 Initialization on SetAddress Command
- 38.4.4.1.4 Initialization on SetConfiguration/SetInterface Command
- 38.4.4.1.5 Endpoint Activation
- 38.4.4.1.6 Endpoint Deactivation
- 38.4.4.1.7 Device DMA/Slave Mode Initialization

38.4.4.1.1 Initialization on USB Reset

1. Set the NAK bit for all OUT endpoints
   - USB_DOEPx_CTL.SNAK = 1 (for all OUT endpoints)
2. Unmask the following interrupt bits:
   - USB_USB_DAINTMSK.INEP0 = 1 (control 0 IN endpoint)
   - USB_USB_DAINTMSK.OUTEP0 = 1 (control 0 OUT endpoint)
   - USB_DOEPMSK.SETUP = 1
   - USB_DOEPMSK.XFERCOMPL = 1
   - USB_DIEPMSK.XFERCOMPL = 1
   - USB_DIEPMSK.TIMEOUTMSK = 1
3. To transmit or receive data, the device must initialize more registers as specified in 38.4.4.1.7 Device DMA/Slave Mode Initialization.
4. Set up the Data FIFO RAM for each of the FIFOs
   - Program the USB_GRXFSIZ Register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 DWORDs (for the status of the control OUT data packet) + 10 DWORDs (for setup packets). If thresholding is enabled, at a minimum, this must be equal to 2 * (USB_DTHRCTL.RXTHRLEN/4 + 1) + 2 DWORDs (for the status of the control OUT data packet) + 10 DWORDs (for setup packets)
   - Program the Device IN Endpoint Transmit FIFO size register (depending on the FIFO number chosen), to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.If thresholding is enabled, this can be programmed to less than one max packet size.
5. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   - USB_DOEP0TSIZ.SUPCNT = 3 (to receive up to 3 back-to-back SETUP packets)
   - In DMA mode, USB_DOEP0DMAADDR register with a memory address to store any SETUP packets received

At this point, all initialization required to receive SETUP packets is done, except for enabling control OUT endpoint 0 in DMA mode.

38.4.4.1.2 Initialization on Enumeration Completion

1. On the Enumeration Done interrupt (USB_GINTSTS.ENUMDONE, read the USB_DSTS register to determine the enumeration speed.
2. Program the USB_DIEP0CTL.MPS field to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. In DMA mode, program the USB_DOEP0CTL register to enable control OUT endpoint 0, to receive a SETUP packet.
   - USB_DOEP0CTL.EPENA = 1

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

38.4.4.1.3 Initialization on SetAddress Command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the USB_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet.

38.4.4.1.4 Initialization on SetConfiguration/SetInterface Command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. For details on a particular endpoint's activation or deactivation, see 38.4.4.1.5 Endpoint Activation and 38.4.4.1.6 Endpoint Deactivation.
5. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the USB_USB_DAINTMSK register.
6. Set up the Data FIFO RAM for each FIFO. See 38.4.5.1 Data FIFO RAM Allocation for more detail.
7. After all required endpoints are configured, the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

38.4.4.1.5 Endpoint Activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the USB_DIEPx_CTL register (for IN or bidirectional endpoints) or the USB_DOEPx_CTL register (for OUT or bidirectional endpoints).
   - Maximum Packet Size
   - USB Active Endpoint = 1
   - Endpoint Start Data Toggle (for interrupt and bulk endpoints)
   - Endpoint Type
   - TxFIFO Number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

38.4.4.1.6 Endpoint Deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB Active Endpoint bit in the USB_DIEPx_CTL register (for IN or bidirectional endpoints) or the USB_DOEPx_CTL register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, resulting in a timeout on the USB.

38.4.4.1.7 Device DMA/Slave Mode Initialization

The application must meed the following conditions to set up the device core to handle traffic.

- In Slave mode, USB_GINTMSK.NPTXFEMPMSK, and USB_GINTMSK.RXFLVLMSK must be unset.
- In DMA mode, the aforementioned interrupts must be masked.

38.4.4.1.8 Transfer Stop Process

When the core is operating as a device, use the following programing sequence if you want to stop any transfers (because of an interrupt from the host, typically a reset).

38.4.4.1.9 Transfer Stop Programming Flow for IN Endpoints

Sequence of operations:

1. Disable the IN endpoint by programming USB_DIEP0CTL/USB_DIEPx_CTL.EPDIS = 1.
2. Wait for the USB_DIEPx_INT.EPDISBLD interrupt, which indicates that the IN endpoint is completely disabled. When the EPDISBLD interrupt is asserted, the core clears the following bits:
   - USB_DIEP0CTL/USB_DIEPx_CTL.EPDIS = 0
   - USB_DIEP0CTL/USB_DIEPx_CTL.EPENA = 0
3. Flush the TX FIFO by programming the following bits:
   - USB_GRSTCTL.TXFFLSH = 1
   - USB_GRSTCTL.TXFNUM = FIFO number specific to endpoint
4. The application can start polling till USB_GRSTCTL.TXFFLSH is cleared. When this bit is cleared, it ensures that there is no data left in the TX FIFO.

38.4.4.1.10 Transfer Stop Programming Flow for OUT Endpoints

Sequence of operations:

1. Enable all OUT endpoints by setting USB_DOEP0CTL/USB_DOEPx_CTL.EPENA = 1.
2. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in 38.4.4.2.2.5 Setting the Global OUT NAK. This ensures that data in the RX FIFO is sent to the application successfully. Set USB_DCTL.USB_DCTL.SGOUTNAK = 1.
3. Wait for the USB_GINTSTS.GOUTNAKEFF interrupt.
4. Disable all active OUT endpoints by programming the following register bits:
   - USB_DOEP0CTL/USB_DOEPx_CTL.EPENA = 1
   - USB_DOEP0CTL/USB_DOEPx_CTL.EPDIS = 1
   - USB_DOEP0CTL/USB_DOEPx_CTL.SNAK = 1
5. Wait for the USB_DOEP0INT/USB_DOEPx_INT.EPDISBLD interrupt for each OUT endpoint programmed in the previous step. The USB_DOEP0INT/USB_DOEPx_INT.EPDISBLD interrupt indicates that the corresponding OUT endpoint is completely disabled. When the EPDISBLD interrupt is asserted, the core clears the following bits:
   - USB_DOEP0CTL/USB_DOEPx_CTL.EPENA = 0
   - USB_DOEP0CTL/USB_DOEPx_CTL.EPDIS = 0

Note: The application must not flush the Rx FIFO, as the Global out nak effective interrupt earlier ensures that there is no data left in the Rx FIFO.

38.4.4.2 Device Programming Operations

38.4.4.2 Device Programming Operations provides links to the programming sequence for different USB transaction types.

Device Mode	IN Without Thresholding	IN With Thresholding	SETUP	OUT Without Thresholding	OUT With Thresholding
Control					
Slave	38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode	Not Supported	38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes	38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes	Not Supported
DMA	38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode	38.4.4.2.3.13 Generic Non-Periodic IN Data Transfers (Bulk and Control) With Thresholding in DMA and Slave Modes	38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes	38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes	38.4.4.2.2.10 Generic Non-Isochronous OUT Data Transfer With Thresholding in DMA and Slave Modes
Bulk					
Slave	38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode	Not Supported		38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes	Not Supported
DMA	38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode	38.4.4.2.3.13 Generic Non-Periodic IN Data Transfers (Bulk and Control) With Thresholding in DMA and Slave Modes		38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes	38.4.4.2.2.10 Generic Non-Isochronous OUT Data Transfer With Thresholding in DMA and Slave Modes
Interrupt					
Slave	38.4.4.2.3.14 Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding and 38.4.4.2.3.15 Generic Periodic IN Data Transfers Without Thresholding Using the Periodic Transfer Interrupt Feature	38.4.4.2.3.16 Generic Periodic (Interrupt and ISO) in Data Transfers With Thresholding		38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes and 38.4.4.2.2.11 Generic Interrupt OUT Data Transfers Without Thresholding Using Periodic Transfer Interrupt Feature	38.4.4.2.2.10 Generic Non-Isochronous OUT Data Transfer With Thresholding in DMA and Slave Modes
DMA	38.4.4.2.3.14 Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding and 38.4.4.2.3.15 Generic Periodic IN Data Transfers Without Thresholding Using the Periodic Transfer Interrupt Feature	38.4.4.2.3.16 Generic Periodic (Interrupt and ISO) in Data Transfers With Thresholding		38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes and 38.4.4.2.2.11 Generic Interrupt OUT Data Transfers Without Thresholding Using Periodic Transfer Interrupt Feature	38.4.4.2.2.10 Generic Non-Isochronous OUT Data Transfer With Thresholding in DMA and Slave Modes
Isochronous					

Slave	38.4.4.2.3.14 Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding	38.4.4.2.3.16 Generic Periodic (Interrupt and ISO) in Data Transfers With Thresholding		38.4.4.2.2.2 Control Read Transfers (SETUP, Data IN, Status OUT), 38.4.4.2.2.12 Generic Isochronous OUT Data Transfer With Thresholding in DMA Mode, and 38.4.4.2.2.14 Incomplete Isochronous OUT Data Transfers in DMA and Slave Modes	38.4.4.2.2.12 Generic Isochronous OUT Data Transfer With Thresholding in DMA Mode
DMA	38.4.4.2.3.14 Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding and 38.4.4.2.3.15 Generic Periodic IN Data Transfers Without Thresholding Using the Periodic Transfer Interrupt Feature	38.4.4.2.3.16 Generic Periodic (Interrupt and ISO) in Data Transfers With Thresholding		38.4.4.2.2.2 Control Read Transfers (SETUP, Data IN, Status OUT), 38.4.4.2.2.12 Generic Isochronous OUT Data Transfer With Thresholding in DMA Mode, and 38.4.4.2.2.14 Incomplete Isochronous OUT Data Transfers in DMA and Slave Modes	38.4.4.2.2.12 Generic Isochronous OUT Data Transfer With Thresholding in DMA Mode

38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes

This section describes the internal data flow and application-level operations during data OUT transfers and setup transactions.

38.4.4.2.1.1 Control Setup Transactions

This section describes how the core handles SETUP packets and the application's sequence for handling setup transactions. To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. See 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

Application Requirements

1. To receive a SETUP packet, the USB_DOEPx_TSIZ.SUPCNT field in a control OUT endpoint must be programmed to a non-zero value. When the application programs the SUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the USB_DOEPx_CTL.NAK status and USB_DOEPx_CTL.EPENA bit setting. The SUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the SUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the SUPCNT field, but the application possibly is not be able to determine the correct number of SETUP packets received in the Setup stage of a control transfer.
   - USB_DOEPx_TSIZ.SUPCNT = 3
2. In DMA mode, the OUT endpoint must also be enabled, to transfer the received SETUP packet data from the internal receive FIFO to the external memory.
   - USB_DOEPx_CTL.EPENA = 1
3. The application must always allocate some extra space in the Receive Data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   - The space to be Reserved is (4 * n) + 6 DWORDs, where n is the number of control endpoints supported by the device. Three DWORDs are required for the first SETUP packet, 1 DWORD is required for the Setup Stage Done DWORD, and 6 DWORDs are required to store two extra SETUP packets among all control endpoints.
   - 3 DWORDs per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (Setup Packet Pattern). The core reserves this space in the receive data
   - FIFO to write SETUP data only, and never uses this space for data packets.
4. In Slave mode, the application must read the 2 DWORDs of the SETUP packet from the receive FIFO. In DMA mode, the core writes the 2 DWORDs of SETUP data to the memory.
5. The application must read and discard the Setup Stage Done DWORD from the receive FIFO.

Internal Data Flow

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and Stall bit settings.
   - The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 DWORDs of data is written to the receive FIFO, and the SUPCNT field is decremented by 1.
   - The first DWORD contains control information used internally by the core
   - The second DWORD contains the first 4 bytes of the SETUP command
   - The third DWORD contains the last 4 bytes of the SETUP command
3. When the Setup stage changes to a Data IN/OUT stage, the core writes an entry (Setup Stage Done DWORD) to the receive FIFO, indicating the completion of the Setup stage.
4. On the AHB side, SETUP packets are emptied either by the DMA or the application. In DMA mode, the SETUP packets (2 DWORDs) are written to the memory location programmed in the USB_DOEPx_DMAADDR register, only if the endpoint is enabled. If the endpoint is not enabled, the data remains in the receive FIFO until the enable bit is set.
5. When either the DMA or the application pops the Setup Stage Done DWORD from the receive FIFO, the core interrupts the application with a USB_DOEPx_INT.SETUP interrupt, indicating it can process the received SETUP packet.
   - The core clears the endpoint enable bit for control OUT endpoints.

Application Programming Sequence

1. Program the USB_DOEPx_TSIZ register.
   - USB_DOEPx_TSIZ.SUPCNT = 3
2. In DMA mode, program the USB_DOEPx_DMAADDR register and USB_DOEPx_CTL register with the endpoint characteristics and set the Endpoint Enable bit (USB_DOEPx_CTL.EPENA).
   - Endpoint Enable = 1
3. In Slave mode, wait for the USB_GINTSTS.RXFLVL interrupt and empty the data packets from the receive FIFO, as explained in 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode. This step can be repeated many times.

4. Assertion of the USB_DOEPx_INT.SETUP interrupt marks a successful completion of the SETUP Data Transfer.
   - On this interrupt, the application must read the USB_DOEPx_TSIZ register to determine the number of SETUP packets received and process the last received SETUP packet.
   - In DMA mode, the application must also determine if the interrupt bit USB_DOEPx_INT.BACK2BACKSETUP is set. This bit is set if the core has received more than three back-to-back SETUP packets. If this is the case, the application must ignore the USB_DOEPx_TSIZ.SUPCNT value and use the USB_DOEPx_DMAADDR directly to read out the last SETUP packet received. USB_DOEPx_DMAADDR-8 provides the pointer to the last valid SETUP data.

Note: If the application has not enabled EP0 before the host sends the SETUP packet, the core ACKs the SETUP packet and stores it in the FIFO, but does not write to the memory until EP0 is enabled. When the application enables the EP0 (first enable) and clears the NAK bit at the same time the Host sends DATA OUT, the DATA OUT is stored in the RxFIFO. The core then writes the setup data to the memory and disables the endpoint. Though the application expects a Transfer Complete interrupt for the Data OUT phase, this does not occur, because the SETUP packet, rather than the DATA OUT packet, enables EP0 the first time. Thus, the DATA OUT packet is still in the RxFIFO until the application re-enables EP0. The application must enable EP0 one more time for the core to process the DATA OUT packet.

Figure 38.17 Processing a SETUP Packet on page 1497 charts this flow.

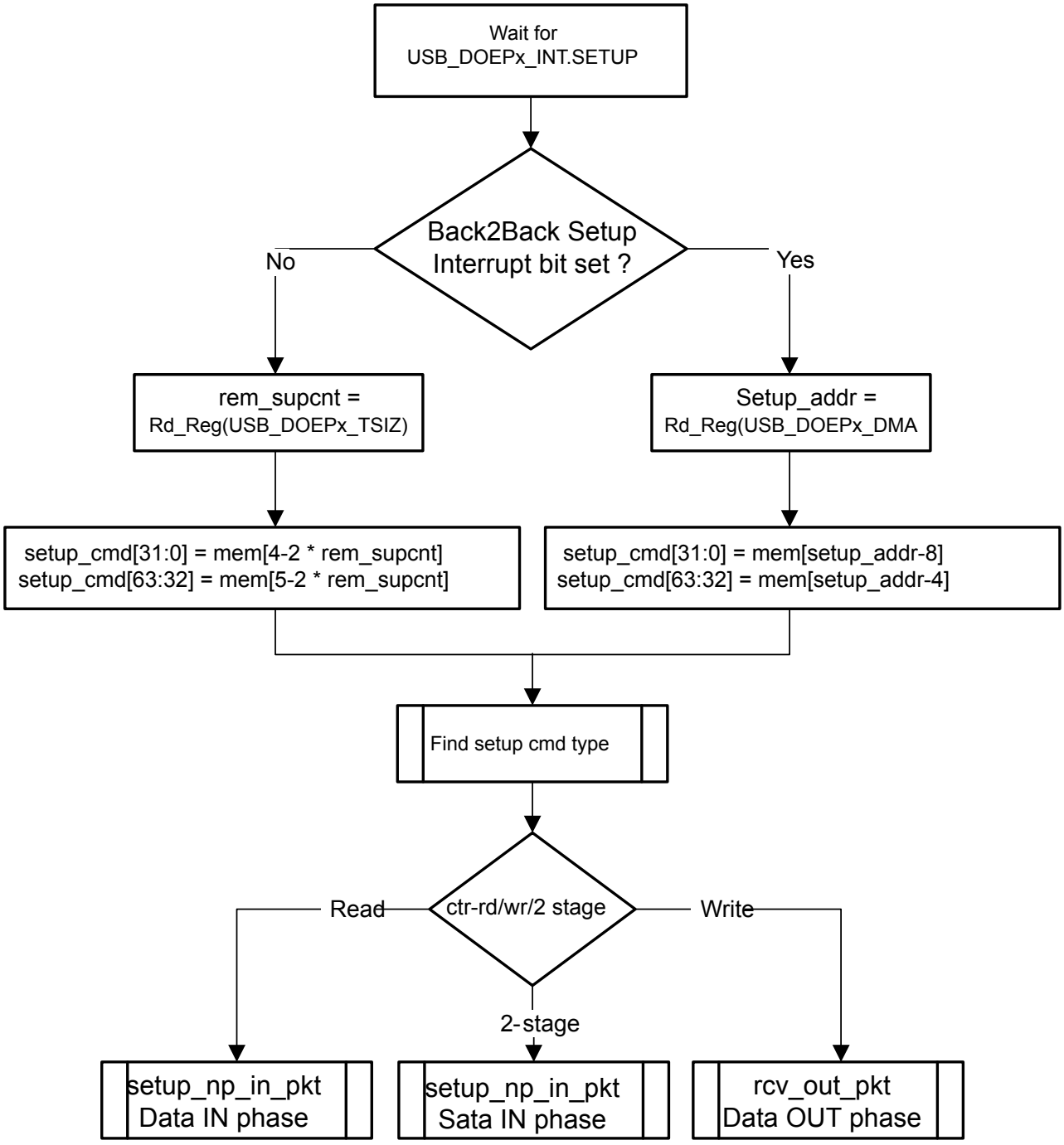


Figure 38.17. Processing a SETUP Packet

38.4.4.2.1.2 Handling More Than Three Back-to-Back SETUP Packets

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the core generates an interrupt (USB_DOEPx_INT.BACK2BACKSETUP). In DMA mode, the core also rewinds the DMA address for that endpoint (USB_DOEPx_DMAADDR) and overwrites the first SETUP packet in system memory with the fourth, second with the fifth, and so on. If the BACK2BACKSETUP interrupt is asserted, the application must read the OUT endpoint DMA register (USB_DOEPx_DMAADDR) to determine the final SETUP data in system memory.

In DMA mode, the application can mask the BACK2BACKSETUP interrupt, but after receiving the DOEPINT.SETUP interrupt, the application can read the DOEPINT.BACK2BACKSETUP interrupt bit. In Slave mode, the application can use the USB_GINTSTS.RXFLVL interrupt to read out the SETUP packets from the FIFO whenever the core receives the SETUP packet.

38.4.4.2.2 Control Transfers

This section describes the various types of control transfers.

38.4.4.2.2.1 Control Write Transfers (SETUP, Data OUT, Status IN)

This section describes control write transfers.

Application Programming Sequence

1. Assertion of the USB_DOEPx_INT.SETUP Packet interrupt indicates that a valid SETUP packet has been transferred to the application. See 38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes for more details. At the end of the Setup stage, the application must reprogram the USB_DOEPx_TSIZ.SUPCNT field to 3 to receive the next SETUP packet.
2. If the last SETUP packet received before the assertion of the SETUP interrupt indicates a data OUT phase, program the core to perform a control OUT transfer as explained in 38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes.

   In DMA mode, the application must reprogram the USB_DOEPx_DMAADDR register to receive a control OUT data packet to a different memory location.
3. In a single OUT data transfer on control endpoint 0, the application can receive up to 64 bytes. If the application is expecting more than 64 bytes in the Data OUT stage, the application must re-enable the endpoint to receive another 64 bytes, and must continue to do so until it has received all the data in the Data stage.
4. Assertion of the USB_DOEPx_INT.Transfer Compl interrupt on the last data OUT transfer indicates the completion of the data OUT phase of the control transfer.
5. On completion of the data OUT phase, the application must do the following.
   - To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint as explained in 38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes.
     - USB_DOEPx_CTL.EPENA = 1
   - To execute the received Setup command, the application must program the required registers in the core. This step is optional, based on the type of Setup command received.
6. For the status IN phase, the application must program the core as described in 38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode to perform a data IN transfer.
7. Assertion of the USB_DIEPx_INT.XFERCOMPL interrupt indicates completion of the status IN phase of the control transfer.
8. The previous step must be repeated until the USB_DIEPx_INT.XFERCOMPL interrupt is detected on the endpoint, marking the completion of the control write transfer.

38.4.4.2.2.2 Control Read Transfers (SETUP, Data IN, Status OUT)

This section describes control read transfers.

Application Programming Sequence

1. Assertion of the USB_DOEPx_INT.SETUP Packet interrupt indicates that a valid SETUP packet has been transferred to the application. See 38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes for more details. At the end of the Setup stage, the application must reprogram the USB_DOEPx_TSIZ.SUPCNT field to 3 to receive the next SETUP packet.
2. If the last SETUP packet received before the assertion of the SETUP interrupt indicates a data IN phase, program the core to perform a control IN transfer as explained in 38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode.
3. On a single IN data transfer on control endpoint 0, the application can transmit up to 64 bytes. To transmit more than 64 bytes in the Data IN stage, the application must re-enable the endpoint to transmit another 64 bytes, and must continue to do so, until it has transmitted all the data in the Data stage.
4. The previous step must be repeated until the USB_DIEPx_INT.XFERCOMPL interrupt is detected for every IN transfer on the endpoint.
5. The USB_DIEPx_INT.XFERCOMPL interrupt on the last IN data transfer marks the completion of the control transfer's Data stage.
6. To perform a data OUT transfer in the status OUT phase, the application must program the core as described in 38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes.
   - The application must program the USB_DCFG.NZSTSOUTHSHK handshake field to a proper setting before transmitting an data OUT transfer for the Status stage.
   - In DMA mode, the application must reprogram the USB_DOEPx_DMAADDR register to receive the control OUT data packet to a different memory location.
7. Assertion of the USB_DOEPx_INT.XFERCOMPL interrupt indicates completion of the status OUT phase of the control transfer. This marks the successful completion of the control read transfer.
   - To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint as explained in 38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes.
   - USB_DOEPx_CTL.EPENA = 1

38.4.4.2.2.3 Two-Stage Control Transfers (SETUP/Status IN)

This section describes two-stage control transfers.

Application Programming Sequence

1. Assertion of the USB_DOEPx_INT.SETUP interrupt indicates that a valid SETUP packet has been transferred to the application. See 38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes for more detail. To receive the next SETUP packet, the application must reprogram the USB_DOEPx_TSIZ.SUPCNT field to 3 at the end of the Setup stage.
2. Decode the last SETUP packet received before the assertion of the SETUP interrupt. If the packet indicates a two-stage control command, the application must do the following.
   - To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint. See 38.4.4.2.1 OUT Data Transfers in Slave and DMA Modes for details.
     - USB_DOEPx_CTL.EPENA = 1
   - Depending on the type of Setup command received, the application can be required to program registers in the core to execute the received Setup command.
3. For the status IN phase, the application must program the core described in 38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode to perform a data IN transfer.
4. Assertion of the USB_DIEPx_INT.XFERCOMPL interrupt indicates the completion of the status IN phase of the control transfer.
5. The previous step must be repeated until the USB_DIEPx_INT.XFERCOMPL interrupt is detected on the endpoint, marking the completion of the two-stage control transfer.

Example: Two-Stage Control Transfer

These notes refer to Figure 38.18 Two-Stage Control Transfer on page 1501.

1. SETUP packet #1 is received on the USB and is written to the receive FIFO, and the core responds with an ACK handshake. This handshake is lost and the host detects a timeout.
2. The SETUP packet in the receive FIFO results in a USB_GINTSTS.RXFLVL interrupt to the application, causing the application to empty the receive FIFO.
3. SETUP packet #2 on the USB is written to the receive FIFO, and the core responds with an ACK handshake.
4. The SETUP packet in the receive FIFO sends the application the USB_GINTSTS.RXFLVL interrupt and the application empties the receive FIFO.
5. After the second SETUP packet, the host sends a control IN token for the status phase. The core issues a NAK response to this token, and writes a Setup Stage Done entry to the receive FIFO. This entry results in a USB_GINTSTS.RXFLVL interrupt to the application, which empties the receive FIFO. After reading out the Setup Stage Done DWORD, the core asserts the USB_DOEPx_INT.SETUP packet interrupt to the application.
6. On this interrupt, the application processes SETUP Packet #2, decodes it to be a two-stage control command, and clears the control IN NAK bit.
   - USB_DIEPx_CTL.CNAK = 1
7. When the application clears the IN NAK bit, the core interrupts the application with a USB_DIEPx_INT.INTKNTXFEMP. On this interrupt, the application enables the control IN endpoint with a USB_DIEPx_TSIZ.XFERSIZE of 0 and a USB_DIEPx_TSIZ.PKTCNT of 1. This results in a zero-length data packet for the status IN token on the USB.
8. At the end of the status IN phase, the core interrupts the application with a USB_DIEPx_INT.XFERCOMPL interrupt.

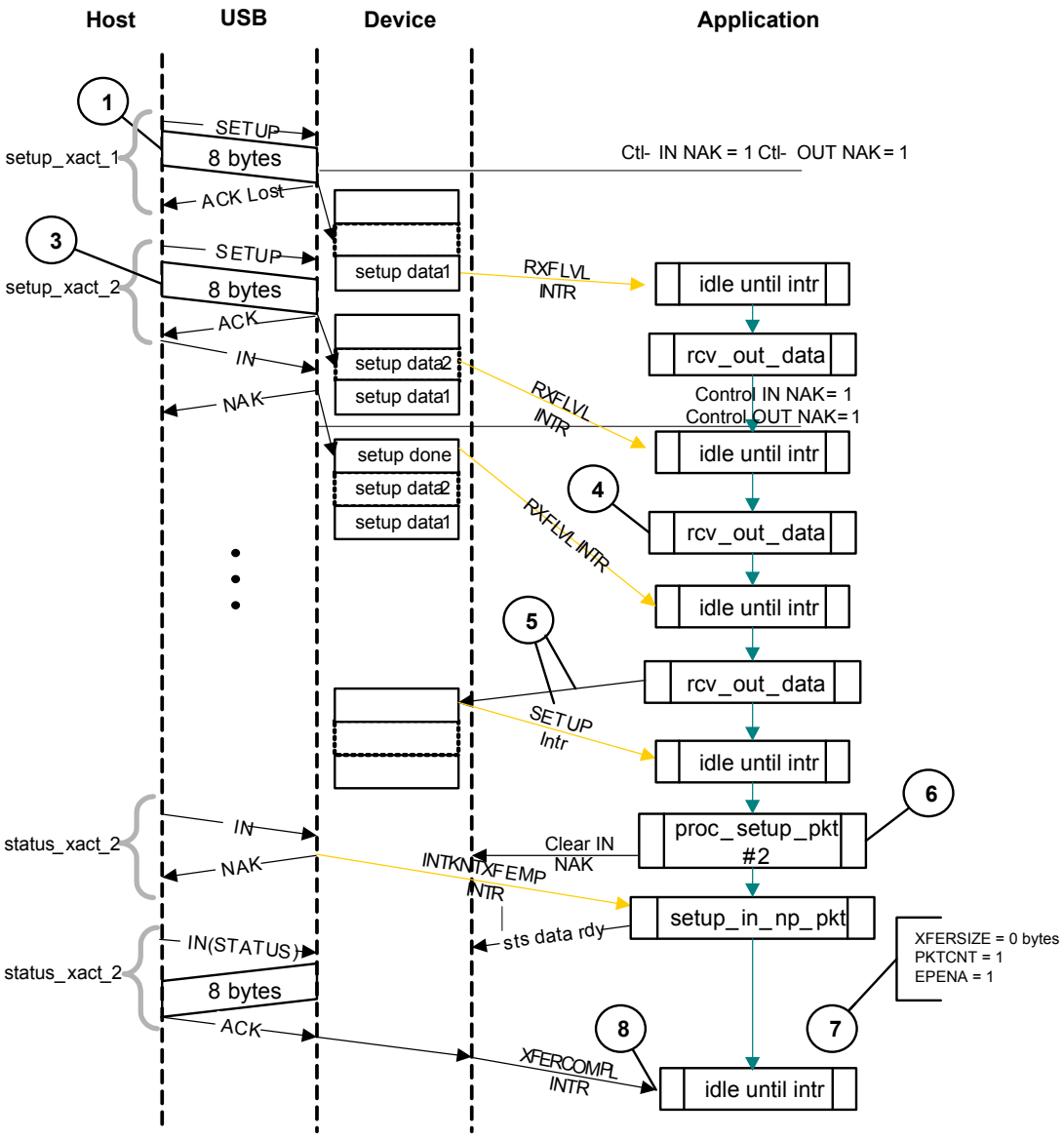


Figure 38.18. Two-Stage Control Transfer

38.4.4.2.2.4 Packet Read From FIFO in Slave Mode

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO in Slave mode.

1. On catching a USB_GINTSTS.RXFLVL interrupt, the application must read the Receive Status Pop register (USB_GRXSTSP).
2. The application can mask the USB_GINTSTS.RXFLVL interrupt by writing to USB_GINTMSK.RXFLVL = 0, until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive Data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the Receive Data FIFO.
4. The receive FIFO's packet status readout indicates one of the following.
5. Global OUT NAK Pattern: PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = Dont Care (0x0), DPID = Dont Care (0b00). This data indicates that the global OUT NAK bit has taken effect.
   a. SETUP Packet Pattern: PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num, DPID = D0. This data indicates that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
   b. Setup Stage Done Pattern: PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = Don't Care (0b00). This data indicates that the Setup stage for the specified endpoint has completed and the Data stage has started. After this entry is popped from the receive FIFO, the core asserts a Setup interrupt on the specified control OUT endpoint.
   c. Data OUT Packet Pattern: PKTSTS = DataOUT, BCNT = size of the Received data OUT packet, EPNUM = EPNum on which the packet was received, DPID = Actual Data PID.
   d. Data Transfer Completed Pattern: PKTSTS = Data OUT Transfer Done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = Dont Care (0b00). This data indicates that a OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a Transfer Completed interrupt on the specified OUT endpoint.

   The encoding for the PKTSTS is listed in 38.7 Register Description.
6. After the data payload is popped from the receive FIFO, the USB_GINTSTS.RXFLVL interrupt must be unmasked.
7. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to USB_GINTSTS.RXFLVL. Reading an empty receive FIFO can result in undefined core behavior.

Figure 38.19 Receive FIFO Packet Read in Slave Mode on page 1502 provides a flow chart of this procedure.

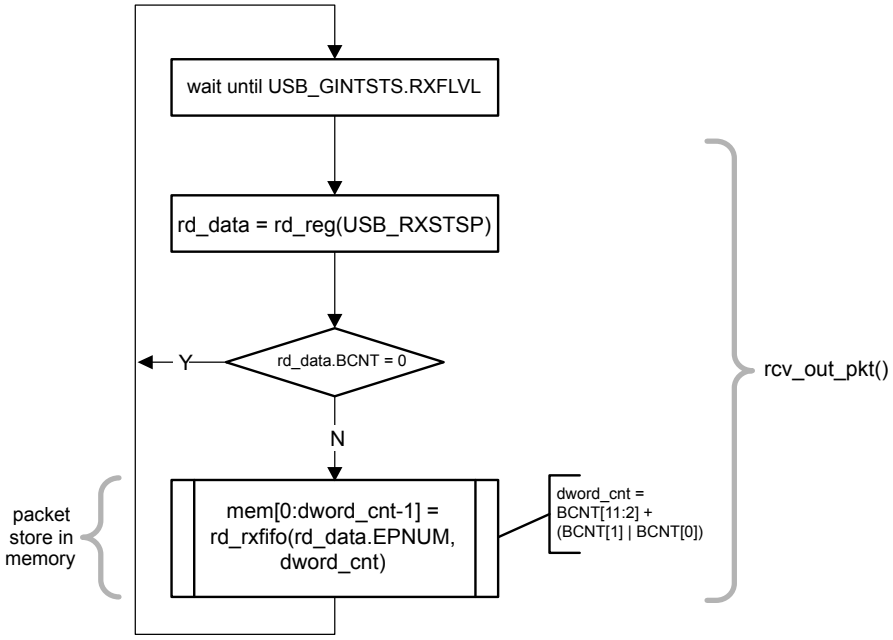


Figure 38.19. Receive FIFO Packet Read in Slave Mode

38.4.4.2.2.5 Setting the Global OUT NAK

Internal Data Flow

1. When the application sets the Global OUT NAK (USB_DCTL.SGOUTNAK), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern. See 38.4.5.1 Data FIFO RAM Allocation.
3. When either the core (in DMA mode) or the application (in Slave mode) pops the Global OUT NAK pattern DWORD from the receive FIFO, the core sets the USB_GINTSTS.GOUTNAKEFF interrupt.
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the USB_DCTL.SGOUTNAK bit.

Application Programming Sequence

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field.
   - USB_DCTL.SGOUTNAK = 1
2. Wait for the assertion of the interrupt USB_GINTSTS.GOUTNAKEFF. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set USB_DCTL.SGOUTNAK and before the core asserts the USB_GINTSTS.GOUTNAKEFF interrupt.
4. The application can temporarily mask this interrupt by writing to the USB_GINTMSK.GOUTNAKEFFMSK bit.
   - USB_GINTMSK.GINNAKEFFMSK = 0
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the USB_DCTL.SGOUTNAK bit. This also clears the USB_GINTSTS.GOUTNAKEFF interrupt.
   - USB_DCTL.CGOUTNAK = 1
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   - USB_GINTMSK.GOUTNAKEFFMSK = 1

38.4.4.2.2.6 Disabling an OUT Endpoint

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application Programming Sequence

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, as described in 38.4.4.2.2.5 Setting the Global OUT NAK.
   - USB_DCTL.SGOUTNAK = 1
   - Wait for the USB_GINTSTS.GOUTNAKEFF interrupt
2. Disable the required OUT endpoint by programming the following fields.
   - USB_DOEPx_CTL.EPDIS = 1
   - USB_DOEPx_CTL.SNAK = 1
3. Wait for the USB_DOEPx_INT.EPDISBLD interrupt, which indicates that the OUT endpoint is completely disabled. When the EPDISBLD interrupt is asserted, the core also clears the following bits.
   - USB_DOEPx_CTL.EPDIS = 0
   - USB_DOEPx_CTL.EPENA = 0
4. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   - USB_DCTL.SGOUTNAK = 0

38.4.4.2.2.7 Stalling a Non-Isochronous OUT Endpoint

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode, as described in 38.4.4.2.2.5 Setting the Global OUT NAK.
2. Disable the required endpoint, as described in 38.4.4.2.2.6 Disabling an OUT Endpoint.
   - When disabling the endpoint, instead of setting the USB_DOEPx_CTL.SNAK bit, set USB_DOEPx_CTL.STALL = 1.
     - The Stall bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the USB_DOEPx_CTL.STALL bit must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the Stall bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. See 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application Requirements

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer, then program that buffer's size and start address (in DMA mode) in the endpoint-specific registers.
1. For OUT transfers, the Transfer Size field in the endpoint's Transfer Size register must be a multiple of the maximum packet size of the endpoint, adjusted to the DWORD boundary.

```
if (mps[epnum] mod 4) == 0
    transfer size[epnum] = n * (mps[epnum])  //Dword Aligned
else
    transfer size[epnum] = n * (mps[epnum] + 4 - (mps[epnum] mod 4)) //Non Dword Aligned

packet count[epnum] = n
n > 0
```

2. In DMA mode, the core stores a received data packet in the memory, always starting on a DWORD boundary. If the maximum packet size of the endpoint is not a multiple of 4, the core inserts byte pads at end of a maximum-packet-size packet up to the end of the DWORD.
3. On any OUT endpoint interrupt, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   - Payload size in memory = application-programmed initial transfer size – core updated final transfer size
   - Number of USB packets in which this payload was received = application-programmed initial packet count – core updated final packet count

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
   - OUT data packets received with Bad Data CRC are flushed from the receive FIFO automatically.
   - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same end-point with the same data PID. In this case the packet count is not decremented.
   - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   - In all the above three cases, the packet count is not decremented because no data is written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data is written to the receive FIFO, either the application (in Slave mode) or the core's DMA engine (in DMA mode), reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
   - The transfer size is 0 and the packet count is 0
   - The last OUT data packet written to the receive FIFO is a short packet (0 <= packet size < maximum packet size)
7. When either the application or the DMA pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application Programming Sequence

1. Program the USB_DOEPx_TSIZ register for the transfer size and the corresponding packet count. Additionally, in DMA mode, program the USB_DOEPx_DMAADDR register.
2. Program the USB_DOEPx_CTL register with the endpoint characteristics, and set the Endpoint Enable and ClearNAK bits.
   - USB_DOEPx_CTL.EPENA = 1
   - USB_DOEPx_CTL.CNAK = 1
3. In Slave mode, wait for the USB_GINTSTS.RXFLVL level interrupt and empty the data packets from the receive FIFO as explained in 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.
   - This step can be repeated many times, depending on the transfer size.
4. Asserting the USB_DOEPx_INT.XFERCOMPL interrupt marks a successful completion of the non-isochronous OUT data transfer.
5. Read the USB_DOEPx_TSIZ register to determine the size of the received data payload.

Note: The XFERSIZE is not decremented for the last packet. This is as per design behavior.

Slave Mode Bulk OUT Transaction

Figure 38.20 Slave Mode Bulk OUT Transaction on page 1505 depicts the reception of a single bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

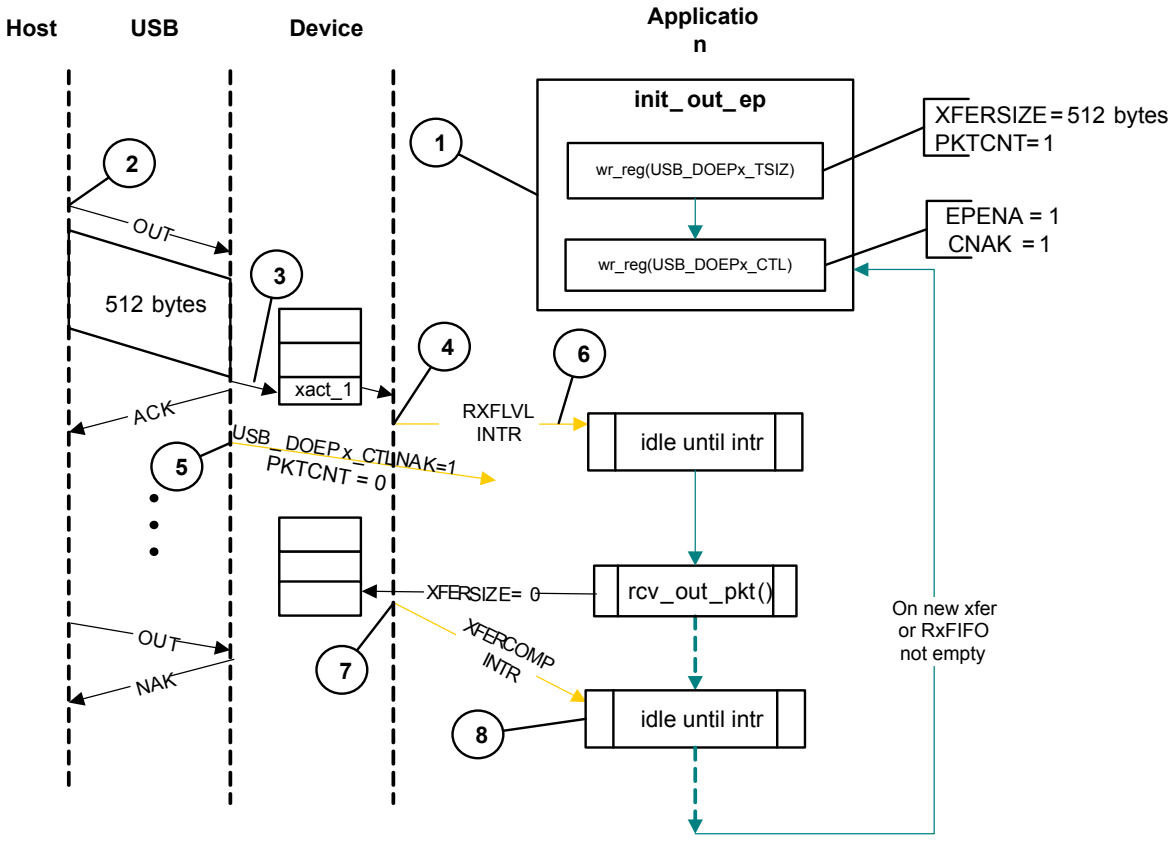


Figure 38.20. Slave Mode Bulk OUT Transaction

After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting USB_DOEPx_CTL.CNAK = 1 and USB_DOEPx_CTL.EPENA = 1, and setting a suitable XFERSIZE and PKTCNT in the USB_DOEPx_TSIZ register.

1. Host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the RxFIFO because space is available there.
3. After writing the complete packet in the RxFIFO, the core then asserts the USB_GINTSTS.RXFLVL interrupt.
4. On receiving the PKTCNT number of USB packets, the core sets the NAK bit for this endpoint internally to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the RxFIFO.
6. When the application has read all the data (equivalent to XFERSIZE), the core generates a USB_DOEPx_INT.XFERCOMPL interrupt.
7. The application processes the interrupt and uses the setting of the USB_DOEPx_INT.XFERCOMPL interrupt bit to determine that the intended transfer is complete.

38.4.4.2.2.9 Generic Isochronous OUT Data Transfer Without Thresholding in DMA and Slave Modes

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. See 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

This section describes a regular isochronous OUT data transfer.

Application Requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers
2. For isochronous OUT data transfers, the Transfer Size and Packet Count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
   - 1 <= packet count[epnum] <= 3
3. In Slave mode, when isochronous OUT endpoints are supported in the device, the application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (USB_GINTSTS.EOPF interrupt). In DMA mode, the application must guarantee enough bandwidth to allow emptying the isochronous OUT data packet from the receive FIFO before the end of each periodic frame.
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the USB_GINTSTS.EOPF and before the USB_GINTSTS.SOF.

Internal Data Flow

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on a isochronous OUT endpoint in a particular frame only if the following condition is met.
   - USB_DOEPx_CTL.DPIDEOF (Even/Odd frame) = USB_DSTS.SOFFN[0]
3. When either the application or the internal DMA completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the USB_DOEPx_TSIZ.RXDPIDSUPCNT (Received DPID) field with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application Programming Sequence

1. Program the USB_DOEPx_TSIZ register for the transfer size and the corresponding packet count. When in DMA mode, also program the USB_DOEPx_DMAADDR register.
2. Program the USB_DOEPx_CTL register with the endpoint characteristics and set the Endpoint Enable, ClearNAK, and Even/Odd frame bits.
   - Endpoint Enable = 1
   - CNAK = 1
   - Even/Odd frame = (0: Even/1: Odd)

1. In Slave mode, wait for the USB_GINTSTS.Rx StsQ level interrupt and empty the data packets from the receive FIFO as explained in 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.
   - This step can be repeated many times, depending on the transfer size.

1. The assertion of the USB_DOEPx_INT.XFERCOMPL interrupt marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory is good.
2. This interrupt can not always be detected for isochronous OUT transfers. Instead, the application can detect the USB_GINTSTS.INCOMPLP (Incomplete Isochronous OUT data) interrupt. See 38.4.4.2.2.14 Incomplete Isochronous OUT Data Transfers in DMA and Slave Modes, for more details
3. Read the USB_DOEPx_TSIZ register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met.
   - USB_DOEPx_TSIZ.RXDPID = D0 and the number of USB packets in which this payload was received = 1
   - USB_DOEPx_TSIZ.RXDPID = D1 and the number of USB packets in which this payload was received = 2
   - USB_DOEPx_TSIZ.RXDPID = D2 and the number of USB packets in which this payload was received = 3
     - The number of USB packets in which this payload was received = App Programmed Initial Packet Count – Core Updated Final Packet Count

The application can discard invalid data packets.

38.4.4.2.2.10 Generic Non-Isochronous OUT Data Transfer With Thresholding in DMA and Slave Modes

This section describes a regular non-ISO OUT data transfer (Control/Bulk/Interrupt) when thresholding is enabled.

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. See 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

Application Requirements:

Application requirements are identical to those for 38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes.

Internal Data Flow:

1. The application must set the transfer size and packet count fields in the endpoint specific registers, clear the NAK bit and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving the data and writes it into the receive FIFO, as long as there is threshold amount of space in the receive FIFO. For every threshold amount of data received on the USB, the data packet and the threshold status are written into the receive FIFO. At the end of a packet, a last threshold status and also a data update status is written into the receive FIFO.If it was the last packet of the transfer, then a transfer complete status is also written into the receive FIFO.On every packet (mps sized or short packet) written into the receive FIFO, the packet count field for that endpoint is decremented by 1.
   - OUT data packets received with Bad Data CRC are flushed out of the receive FIFO automatically.The core also rewind the DMA pointers internally.
   - Non-ISO OUT data packet re-sent by the host, because the ACK was not seen by the host, is discarded by the core, after sending an ACK for the packet on the USB. The application does not see multiple back-to-back data OUT packets on the same endpoint, with the same data PID. In this case the packet count is not decremented.
   - If there is no space for at least threshold amount of data in the receive FIFO, the ISO/non-ISO data packets are ignored and not written into the receive FIFO. In addition, the non-ISO OUT tokens are responded with NAK handshake.
   - If the core sees an overflow case (no space in the fifo in the middle of a packet reception), then the core stops writing the remaining data into the fifo and sends a NAK handshake on the USB. The core rewinds the fifo pointer to the threshold boundary, so that the portion of the threshold data that is in the fifo is flushed out.The core also rewinds the DMA pointers.The core also sets USB_DOEPx_INT.OUTPKTERR (This interrupt bit is mainly used for debugging).

   In all the above cases, the packet count is not decremented because no data is written into the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the ISO/non-ISO data packets are ignored and not written into the receive FIFO and the non-ISO OUT tokens are responded with a NAK handshake.
4. The DMA engine transfers data from the receive FIFO to the system memory as soon as it sees one threshold amount of data in the FIFO.
5. At the end of every packet write on the AHB into the external memory, the transfer size for the endpoint is decremented by the size of packet written into the memory.
6. The OUT Data Transfer Complete pattern for an OUT endpoint is written into the receive FIFO, on one of the following conditions.
   - The last threshold is written into FIFO and the packet count is decremented to 0
   - If it is a short packet and the core sees the end of packet within a threshold.
7. When this entry (OUT Data Transfer Complete) is popped out by the DMA engine, Transfer Complete interrupt for the endpoint is generated and the endpoint enable is cleared.
8. “Rewind OUT Data Transfer” pattern is written into the receive FIFO, on one of the following conditions
   - On seeing Overflow condition.
   - On seeing a CRC error.
9. When this entry (Rewind OUT Data Transfer) is popped out by the DMA engine, it does the DMA pointer rewind.

Application Programming Sequence

This sequence is identical to that described for 38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes.

38.4.4.2.2.11 Generic Interrupt OUT Data Transfers Without Thresholding Using Periodic Transfer Interrupt Feature

This section describes a regular INTR OUT data transfer with the Periodic Transfer Interrupt feature.

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. See 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

Application Requirements

1. Before setting up a periodic OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer, then program that buffer's size and start address in the endpoint-specific registers.
2. For Interrupt OUT transfers, the Transfer Size field in the endpoint's Transfer Size register must be a multiple of the maximum packet size of the endpoint, adjusted to the DWORD boundary. The Transfer Size programmed can span across multiple frames based on the periodicity after which the application want to receive the USB_DOEPx_INT.XFERCOMPL interrupt
   - transfer size[epnum] = n * (mps[epnum] + 4 - (mps[epnum] mod 4))
   - packet count[epnum] = n
   - n > 0 (Higher value of n reduces the periodicity of the USB_DOEPx_INT.XFERCOMPL interrupt)
   - 1 < packet count[epnum] < n (Higher value of n reduces the periodicity of the USB_DOEPx_INT.XFERCOMPL interrupt)
3. In DMA mode, the core stores a received data packet in the memory, always starting on a DWORD boundary. If the maximum packet size of the endpoint is not a multiple of 4, the core inserts byte pads at end of a maximum-packet-size packet up to the end of the DWORD. The application will not be informed about the frame number on which a specific packet has been received.
4. On USB_DOEPx_INT.XFERCOMPL interrupt, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   - Payload size in memory = application-programmed initial transfer size – core updated final transfer size
   - Number of USB packets in which this payload was received = application-programmed initial packet count – core updated final packet count.
   - If for some reason, the host stops sending tokens, there are no interrupts to the application, and the application must timeout on its own.
5. The assertion of the USB_DOEPx_INT.XFERCOMPL interrupt marks the completion of the interrupt OUT data transfer. This interrupt does not necessarily mean that the data in memory is good.
6. Read the USB_DOEPx_TSIZ register to determine the size of the received transfer and to determine the validity of the data received in the frame.

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
   - The application must enable the USB_DCTL.IGNRFRMNUM
2. When an interrupt OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame will be ignored by the core.

1. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
   - OUT data packets received with Bad Data CRC or any packet error are flushed from the receive FIFO automatically.
   - Interrupt packets with PID errors are not passed to application. Core discards the packet, sends ACK and does not decrement packet count.
   - If there is no space in the receive FIFO, interrupt data packets are ignored and not written to the receive FIFO. Additionally, interrupt OUT tokens receive a NAK handshake reply.
2. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or interrupt data packets are ignored and not written to the receive FIFO, and interrupt OUT tokens receive a NAK handshake reply.
3. After the data is written to the receive FIFO, the core's DMA engine reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
4. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
5. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
   - The transfer size is 0 and the packet count is 0.
   - The last OUT data packet written to the receive FIFO is a short packet (0 < packet size < maximum packet size)
6. When either the application or the DMA pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint and the endpoint enable is cleared.

38.4.4.2.2.12 Generic Isochronous OUT Data Transfer With Thresholding in DMA Mode

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. See 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

This section describes a regular isochronous OUT data transfer.

Application Requirements:

Application requirements are identical to those for 38.4.4.2.2.8 Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes.

Internal Data Flow:

1. The internal data flow for isochronous OUT Endpoints when thresholding is enabled, is the same as that for the non isochronous OUT endpoints when thresholding is enabled, but for a few differences.
2. If MAC sees an overflow condition when writing a packet, it stops writing.The current threshold amount of data that is being written into the receive FIFO is flushed out at the end of packet.This eventually results in USB_GINTSTS.INCOMPLP (Incomplete ISO OUT Data) interrupt.
3. If MAC sees a CRC error for the receiving packet, the last threshold being written into the receive FIFO is flushed out. This eventually results in USB_GINTSTS.INCOMPLP (Incomplete isochronous OUT Data) interrupt.
4. Assertion of USB_DOEPx_INT.XFERCOMPL interrupt marks a completion of the isochronous OUT Data Transfer. This interrupt may not necessarily mean that data in the memory is good data.
5. Read the USB_DOEPx_TSIZ register, to find out the size of the received transfer and to find out the validity of the data received in the frame. The application must treat the data received into the memory as valid only if one of the following conditions is met. Invalid data packets may be discarded by the application.
   - USB_DOEPx_TSIZ.RXDPID = D0 and Number of USB Packets in which this payload was received = 1
   - USB_DOEPx_TSIZ.RXDPID = D1 and Number of USB Packets in which this payload was received = 2
   - USB_DOEPx_TSIZ.RXDPID = D2 and Number of USB Packets in which this payload was received = 3

   Number of USB Packets in which this payload was received = App Programmed Initial Packet Count - Core Updated Final Packet Count

38.4.4.2.2.13 Generic Isochronous OUT Data Transfers Without Thresholding Using Periodic Transfer Interrupt Feature

This section describes a regular isochronous OUT data transfer with the Periodic Transfer Interrupt feature.

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. For packet writes in Slave mode, see: 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

Application Requirements

1. Before setting up ISOC OUT transfers spanned across multiple frames, the application must allocate buffer in the memory to accommodate all data to be received as part of the OUT transfers, then program that buffer's size and start address in the endpoint-specific registers.
   - The application must mask the USB_GINTSTS.INCOMPLP (Incomplete ISO OUT).
   - The application must enable the USB_DCTL.IGNRFRMNUM
2. For ISOC transfers, the Transfer Size field in the USB_DOEPx_TSIZ.XFERSIZE register must be a multiple of the maximum packet size of the endpoint, adjusted to the DWORD boundary. The Transfer Size programmed can span across multiple frames based on the periodicity after which the application wants to receive the USB_DOEPx_INT.XFERCOMPL interrupt
   - transfer size[epnum] = n * (mps[epnum] + 4 - (mps[epnum] mod 4))
   - packet count[epnum] = n
   - n > 0 (Higher value of n reduces the periodicity of the USB_DOEPx_INT.XFERCOMPL interrupt)
   - 1 =< packet count[epnum] =< n (Higher value of n reduces the periodicity of the USB_DOEPx_INT.XFERCOMPL interrupt).
3. In DMA mode, the core stores a received data packet in the memory, always starting on a DWORD boundary. If the maximum packet size of the endpoint is not a multiple of 4, the core inserts byte pads at end of a maximum-packet-size packet up to the end of the DWORD. The application will not be informed about the frame number and the PID value on which a specific OUT packet has been received.
4. The assertion of the USB_DOEPx_INT.XFERCOMPL interrupt marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory is good.
   - On USB_DOEPx_INT.XFERCOMPL, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory.
   - Payload size in memory = application-programmed initial transfer size - core updated final transfer size
   - Number of USB packets in which this payload was received = application-programmed initial packet count – core updated final packet count.
   - If for some reason, the host stop sending tokens, there will be no interrupt to the application, and the application must timeout on its own.
5. The assertion of the USB_DOEPx_INT.XFERCOMPL can also mark a packet drop on USB due to unavailability of space in the RxFifo or due to any packet errors.
   - The application must read the USB_DOEPx_INT.PKTDRPSTS (USB_DOEPx_INT.Bit[11] is now used as the USB_DOEPx_INT.PKTDRPSTS) register to differentiate whether the USB_DOEPx_INT.XFERCOMPL was generated due to the normal end of transfer or due to dropped packets. In case of packets being dropped on the USB due to unavailability of space in the RxFifo or due to any packet errors the endpoint enable bit is cleared.
   - In case of packet drop on the USB application must re-enable the endpoint after recalculating the values USB_DOEPx_TSIZ.XFERSIZE and USB_DOEPx_TSIZ.PKTCNT.
   - Payload size in memory = application-programmed initial transfer size - core updated final transfer size
   - Number of USB packets in which this payload was received = application-programmed initial packet count - core updated final packet count.

Note: Due to application latencies it is possible that DOEPINT.XFERCOMPL interrupt is generated without DOEPINT.PKTDRPSTS being set, This scenario is possible only if back-to-back packets are dropped for consecutive frames and the PKTDRPSTS is merged, but the XFERSIZE and PktCnt values for the endpoint are nonzero. In this case, the application must proceed further by programming the PKTCNT and XFERSIZE register for the next frame, as it would if PKTDRPSTS were being set.

Figure 38.21 ISOC OUT Application Flow for Periodic Transfer Interrupt Feature on page 1511 gives the application flow for Isochronous OUT Periodic Transfer Interrupt feature.

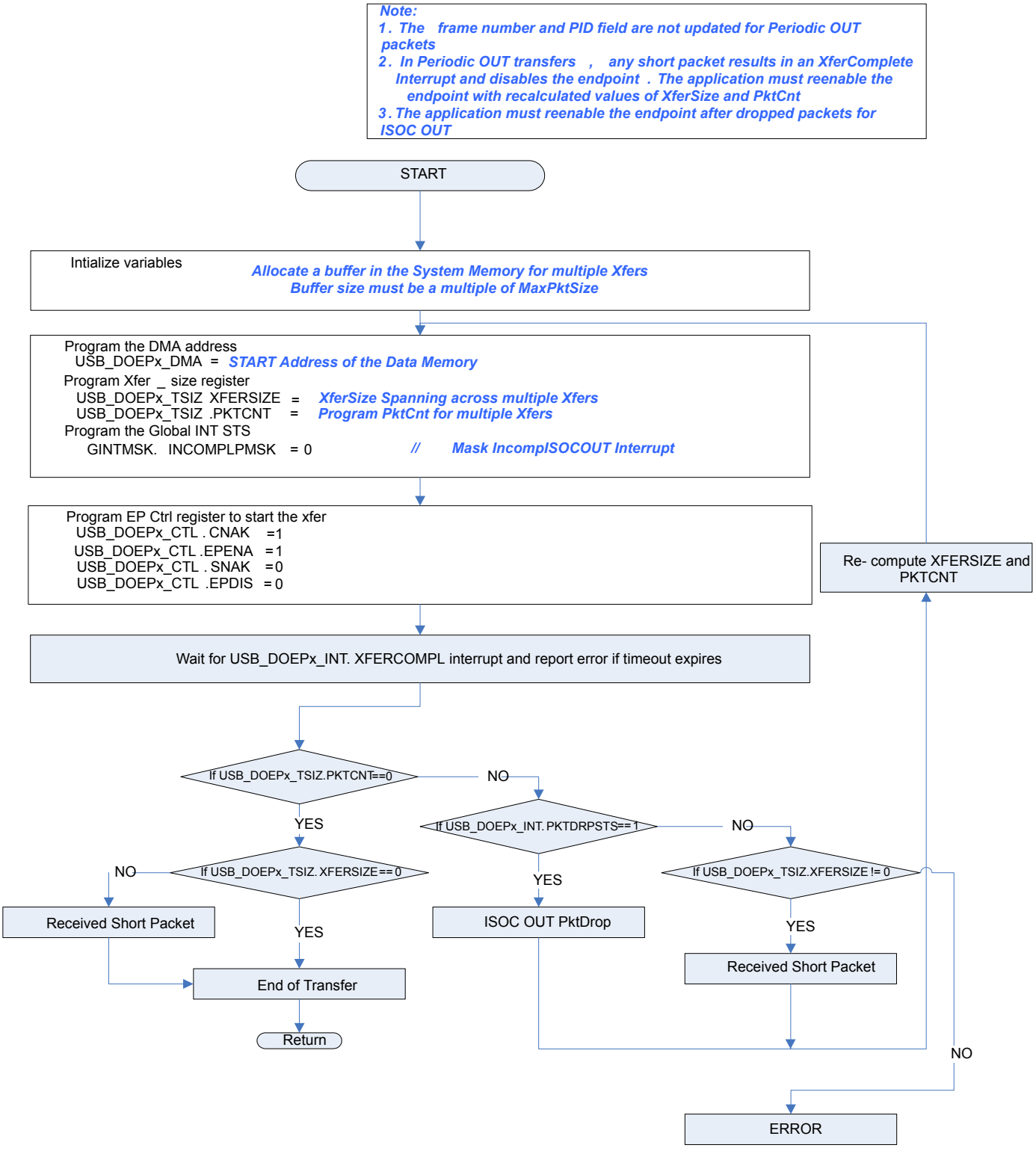


Figure 38.21. ISOC OUT Application Flow for Periodic Transfer Interrupt Feature

Internal Data Flow

1. The application must set the Transfer Size, Packets to be received in a frame and Packet Count Fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame will be ignored by the core.

3. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
4. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the ISOC packets are ignored and not written to the receive FIFO.
5. After the data is written to the receive FIFO, the core's DMA engine, reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
6. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
7. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
   - The transfer size is 0 and the packet count is 0
   - The last OUT data packet written to the receive FIFO is a short packet (0 < packet size < maximum packet size).
8. When the DMA pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint or the endpoint enable is cleared.
9. OUT data packets received with Bad Data CRC or any packet error are flushed from the receive FIFO automatically.
   - In these two cases, the packet count and transfer size registers are not decremented because no data is written to the receive FIFO.

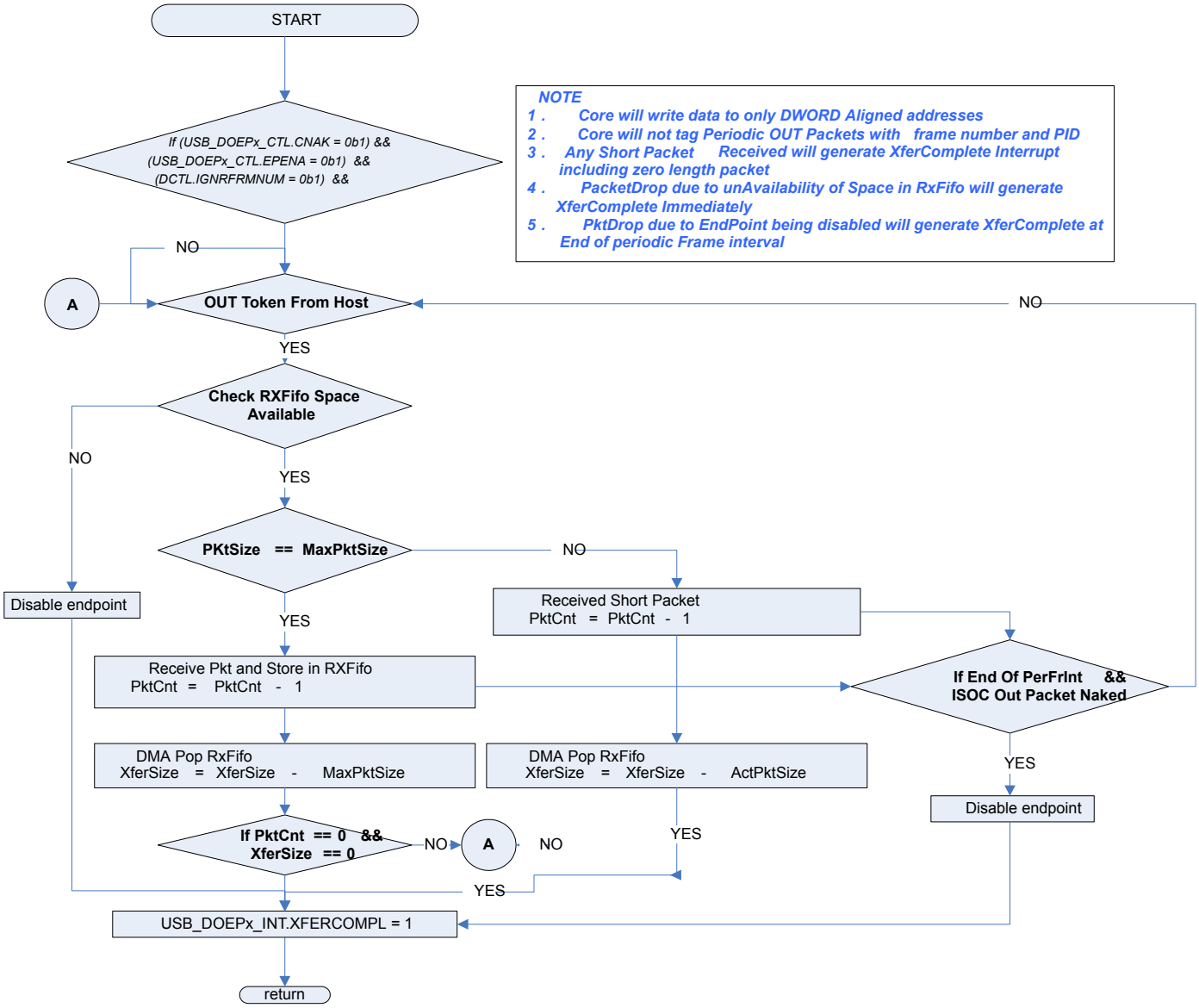


Figure 38.22. Isochronous OUT Core Internal Flow for Periodic Transfer Interrupt Feature

38.4.4.2.2.14 Incomplete Isochronous OUT Data Transfers in DMA and Slave Modes

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. See 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal Data Flow

1. For isochronous OUT endpoints, the USB_DOEPx_INT.XFERCOMPL interrupt possibly is not always asserted. If the core drops isochronous OUT data packets, the application could fail to detect the USB_DOEPx_INT.XFERCOMPL interrupt under the following circumstances.
   - When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data. In thresholding this is same as overflow.
   - When the isochronous OUT data packet is received with CRC errors
   - When the isochronous OUT token received by the core is corrupted
   - When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the USB_GINTSTS.INCOMPLP (Incomplete Isochronous OUT data) interrupt, indicating that a USB_DOEPx_INT.XFERCOMPL interrupt is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remains in progress on this endpoint on the USB.
3. This step is applicable only if the core is operating in slave mode. Application Programming Sequence
4. This step is applicable only if the core is operating in slave mode. Asserting the USB_GINTSTS.INCOMPLP (Incomplete Isochronous OUT data) interrupt indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
5. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the DMA or the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   - When all data is emptied from the receive FIFO, the application can detect the USB_DOEPx_INT.XFERCOMPL interrupt. In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame, as described in 38.4.4.2.2.2 Control Read Transfers (SETUP, Data IN, Status OUT).
6. When it receives a USB_GINTSTS.incomplete Isochronous OUT data interrupt, the application must read the control registers of all isochronous OUT endpoints (USB_DOEPx_CTL) to determine which endpoints had an incomplete transfer in the current frame. An endpoint transfer is incomplete if both the following conditions are met.
   - USB_DOEPx_CTL.DPIDEOF (Even/Odd frame) = USB_DSTS.SOFFN[0]
   - USB_DOEPx_CTL.EPENA (Endpoint Enable) = 1
7. The previous step must be performed before the USB_GINTSTS.SOF interrupt is detected, to ensure that the current frame number is not changed.
8. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the USB_DOEPx_CTL.EPDIS (Endpoint Disable) bit.
9. Wait for the USB_DOEPx_INT.EPDIS (Endpoint Disabled) interrupt and enable the endpoint to receive new data in the next frame as explained in 38.4.4.2.2.2 Control Read Transfers (SETUP, Data IN, Status OUT).
   - Because the core can take some time to disable the endpoint, the application possibly is not able to receive the data in the next frame after receiving bad isochronous data.

38.4.4.2.3 IN Data Transfers in Slave and DMA Modes

This section describes the internal data flow and application-level operations during IN data transfers.

38.4.4.2.3.1 Packet Write in Slave Mode

This section describes how the application writes data packets to the endpoint FIFO in Slave mode.

1. The application can either choose polling or interrupt mode.
   - In polling mode, application monitors the status of the endpoint transmit data FIFO, by reading the USB_DIEPx_TXFSTS register, to determine, if there is enough space in the data FIFO.
   - In interrupt mode, application waits for the USB_DIEPx_INT.TXFEMP interrupt and then reads the USB_DIEPx_TXFSTS register, to determine, if there is enough space in the data FIFO.
   - To write a single non-zero length data packet, there must be space to write the entire packet is the data FIFO.
   - For writing zero length packet, application must not look for FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. The application, typically must do a read modify write on the USB_DIEPx_CTL, to avoid modifying the contents of the register, except for setting the Endpoint Enable bit.

The application can write multiple packets for the same endpoint, into the transmit FIFO, if space is available. For periodic IN endpoints, application must write packets only for one frame. It can write packets for the next periodic transaction, only after getting transfer complete for the previous transaction.

38.4.4.2.3.2 Setting Global Non-Periodic in Endpoint NAK

Internal Data Flow

1. When the application sets the Global Non-periodic IN NAK bit (USB_DCTL.SGNPINNAK), the core stops transmitting data on the non-periodic endpoint, irrespective of data availability in the Non-periodic Transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
3. The core asserts the USB_GINTSTS.GINNAKEFF interrupt in response to the USB_DCTL.SGNPINNAK bit.
4. Once the application detects this interrupt, it can assume that the core is in the Global Non-periodic IN NAK mode. The application can clear this interrupt by clearing the USB_DCTL.SGNPINNAK bit.

Application Programming Sequence

1. To stop transmitting any data on non-periodic IN endpoints, the application must set the USB_DCTL.SGNPINNAK bit. To set this bit, the following field must be programmed
   - USB_DCTL.SGNPINNAK = 1
2. Wait for the assertion of the USB_GINTSTS.GINNAKEFF interrupt. This interrupt indicates the core has stopped transmitting data on the non-periodic endpoints.
3. The core can transmit valid non-periodic IN data after the application has set the USB_DCTL.SGNPINNAK bit, but before the assertion of the USB_GINTSTS.GINNAKEFF interrupt.
4. The application can optionally mask this interrupt temporarily by writing to the USB_GINTMSK.GINNAKEFFMSK bit.
   - USB_GINTMSK.GINNAKEFFMSK = 0
5. To exit Global Non-periodic IN NAK mode, the application must clear the USB_DCTL.SGNPINNAK. This also clears the USB_GINTSTS.GINNAKEFF interrupt.
   - USB_DCTL.SGNPINNAK = 1
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   - USB_GINTMSK.GINNAKEFFMSK = 1

38.4.4.2.3.3 Setting IN Endpoint NAK

Internal Data Flow

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   - Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the USB_DIEPx_INT.INEPNAKEFF (IN NAK Effective) interrupt in response to the USB_DIEPx_CTL.SNAK (Set NAK) bit.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the USB_DIEPx_CTL. Clear NAK bit.

Application Programming Sequence

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   - USB_DIEPx_CTL.SNAK = 1
2. Wait for assertion of the USB_DIEPx_INT.INEPNAKEFF (NAK Effective) interrupt. This interrupt indicates the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the USB_DIEPMSK.INEPNAKEFFMSK (NAK Effective) bit.
   - USB_DIEPMSK.INEPNAKEFFMSK (NAK Effective) = 0
5. To exit Endpoint NAK mode, the application must clear the USB_DIEPx_CTL.NAK status. This also clears the USB_DIEPx_INT.INEPNAKEFF (NAK Effective) interrupt.
   - USB_DIEPx_CTL.CNAK = 1
6. If the application masked this interrupt earlier, it must be unmasked as follows:
   - USB_DIEPMSK.INEPNAKEFFMSK (NAK Effective) = 1

38.4.4.2.3.4 IN Endpoint Disable

Use the following sequence to disable a specific IN endpoint (periodic/non-periodic) that has been previously enabled.

Application Programming Sequence:

1. In Slave mode, the application must stop writing data on the AHB, for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode. See 38.4.4.2.3.3 Setting IN Endpoint NAK.
   - USB_DIEPx_CTL.SNAK = 1
3. Wait for USB_DIEPx_INT.INEPNAKEFF (NAK Effective) interrupt.
4. Set the following bits in the USB_DIEPx_CTL register for the endpoint that must be disabled.
   - USB_DIEPx_CTL.EPDIS (Endpoint Disable) = 1
   - USB_DIEPx_CTL.SNAK = 1
5. Assertion of USB_DIEPx_INT.EPDISBLD (Endpoint Disabled) interrupt indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits.
   - USB_DIEPx_CTL.EPENA = 0
   - USB_DIEPx_CTL.EPDIS = 0
6. The application must read the USB_DIEPx_TSIZ register for the periodic IN EP, to calculate how much data on the endpoint was transmitted on the USB.
7. The application must flush the data in the Endpoint transmit FIFO, by setting the following fields in the USB_GRSTCTL register.
   - USB_GRSTCTL.TXFNUM = Endpoint Transmit FIFO Number
   - USB_GRSTCTL.TXFFLSH = 1

The application must poll the USB_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

38.4.4.2.3.5 Bulk IN Stall

These notes refer to Figure 38.23 Bulk IN Stall on page 1517

1. The application has scheduled an IN transfer on receiving the USB_DIEPx_INT.INTKNTXFEMP (IN Token Received When TxFIFO Empty) interrupt.
2. When the transfer is in progress, the application must force a STALL on the endpoint. This could be because the application has received a SetFeature.Endpoint Halt command. The application sets the Stall bit, disables the endpoint and waits for the USB_DIEPx_INT.EPDISBLD (Endpoint Disabled) interrupt. This generates STALL handshakes for the endpoint on the USB.
3. On receiving the interrupt, the application flushes the Non-periodic Transmit FIFO and clears the USB_DCTL.SGNPINNAK (Global IN NP NAK) bit.
4. On receiving the ClearFeature.Endpoint Halt command, the application clears the Stall bit.
5. The endpoint behaves normally and the application can re-enable the endpoint for new transfers

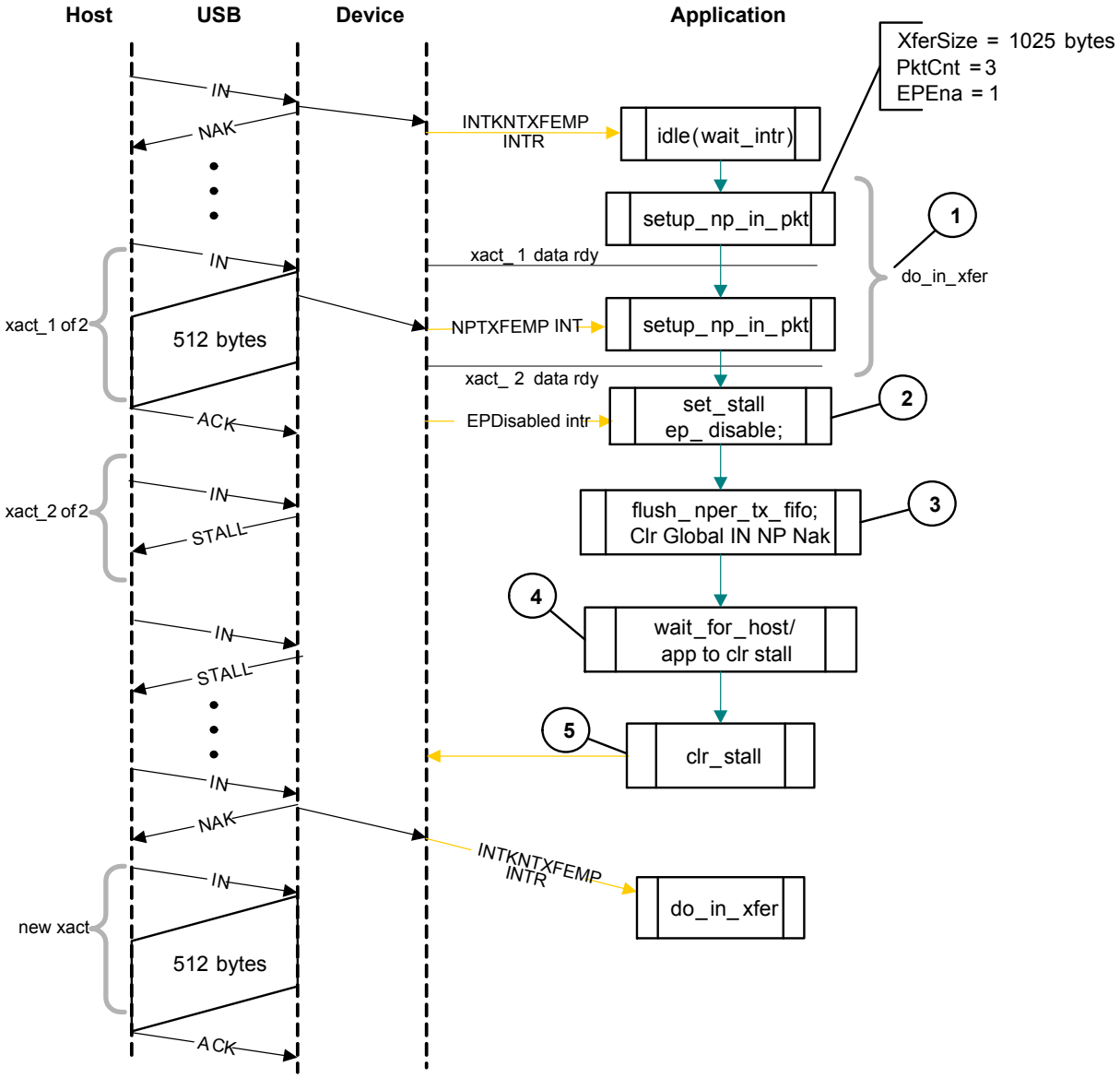


Figure 38.23. Bulk IN Stall

38.4.4.2.3.6 Incomplete Isochronous IN Data Transfers

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal Data Flow

1. An isochronous IN transfer is treated as incomplete in one of the following conditions.
   a. The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects a USB_GINTSTS.INCOMPISOIN (Incomplete Isochronous IN Transfer) interrupt.
   b. The application or DMA is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects a USB_DIEPx_INT.INTKNTXFEMP (IN Token Received When TxFIFO Empty) interrupt. The application can ignore this interrupt, as it eventually results in a USB_GINTSTS.INCOMPISOIN (Incomplete Isochronous IN Transfer) interrupt at the end of periodic frame.
      i. The core transmits a zero-length data packet on the USB in response to the received IN token.
   c. If thresholding is enabled and there was an underrun condition, the core also generates a USB_DIEPx_INT.TXFIFOUNDRN interrupt. The application can ignore this interrupt, which eventually results in a USB_GINTSTS.INCOMPISOIN (Incomplete ISO IN Transfer) interrupt.
2. In either of the aforementioned cases, in Slave mode, the application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint. In DMA mode, the core automatically stops fetching the data payload when the endpoint disable bit is set.
4. The core disables the endpoint, clears the disable bit, and asserts the Endpoint Disable interrupt for the endpoint.

Application Programming Sequence

1. The application can ignore the USB_DIEPx_INT.INTKNTXFEMP (IN Token Received When TxFIFO empty) interrupt on any isochronous IN endpoint, as it eventually results in a USB_GINTSTS.INCOMPISOIN (Incomplete Isochronous IN Transfer) interrupt. The application can also ignore USB_DIEPx_INT.TXFIFOUNDRN interrupt when thresholding is enabled.
2. Assertion of the USB_GINTSTS.INCOMPISOIN (Incomplete Isochronous IN Transfer) interrupt indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the Endpoint Control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. In Slave mode, the application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. In both modes of operation, program the following fields in the USB_DIEPx_CTL register to disable the endpoint.
   - USB_DIEPx_CTL.SNAK = 1
   - USB_DIEPx_CTL.EPDIS (Endpoint Disable) = 1
6. The USB_DIEPx_INT.EPDISBLD (Endpoint Disabled) interrupt's assertion indicates that the core has disabled the endpoint.
   - At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next frame. To flush the data, the application must use the USB_GRSTCTL register.

38.4.4.2.3.7 Stalling Non-Isochronous IN Endpoints

This section describes how the application can stall a non-isochronous endpoint.

Application Programming Sequence

1. Disable the IN endpoint to be stalled. Set the Stall bit as well.
2. USB_DIEPx_CTL.EPDIS (Endpoint Disable) = 1, when the endpoint is already enabled
   - USB_DIEPx_CTL.STALL = 1
   - The Stall bit always takes precedence over the NAK bit
3. Assertion of the USB_DIEPx_INT.EPDISBLD (Endpoint Disabled) interrupt indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the Non-periodic or Periodic Transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints, which do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the USB_DIEPx_CTL.STALL bit must be cleared.
6. If the application sets or clears a STALL for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the Stall bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

Special Case: Stalling the Control IN/OUT Endpoint

The core must stall IN/OUT tokens if, during the Data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must to enable USB_DIEPx_INT.INTKNTXFEMP and USB_DOEPx_INT.OUTTKNEPDIS interrupts during the Data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

38.4.4.2.3.8 Worst-Case Response Time

When the acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token could come sooner. This worst case value is 7 PHY clocks in FS mode.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the INCOMPISOIN and INCOMPLP interrupts inform the application that isochronous IN/OUT packets were dropped.

38.4.4.2.3.9 Choosing the Value of USB_GUSBCFG.USBTRDTIM

The value in USB_GUSBCFG.USBTRDTIM is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from PFC (Packet FIFO Controller) block. This time involves the synchronization delay between the PHY and AHB clocks. This delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes it into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

If the AHB is running at a higher frequency than the PHY (in Low-speed mode), the application can use a smaller value for USB_GUSBCFG.USBTRDTIM. Figure 38.24 USBTRDTIM Max Timing Case ERROR Wrong Image on page 1521 explains the 5-clock delay. This diagram has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from pclk to hclk domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

The application can use the following formula to calculate the value of USB_GUSBCFG.USBTRDTIM:

4 * AHB Clock + 1 PHY Clock = (2 clock sync + 1 clock memory address + 1 clock memory data from sync RAM) + (1 PHY Clock (next PHY clock MAC can sample the 2-clock FIFO output)

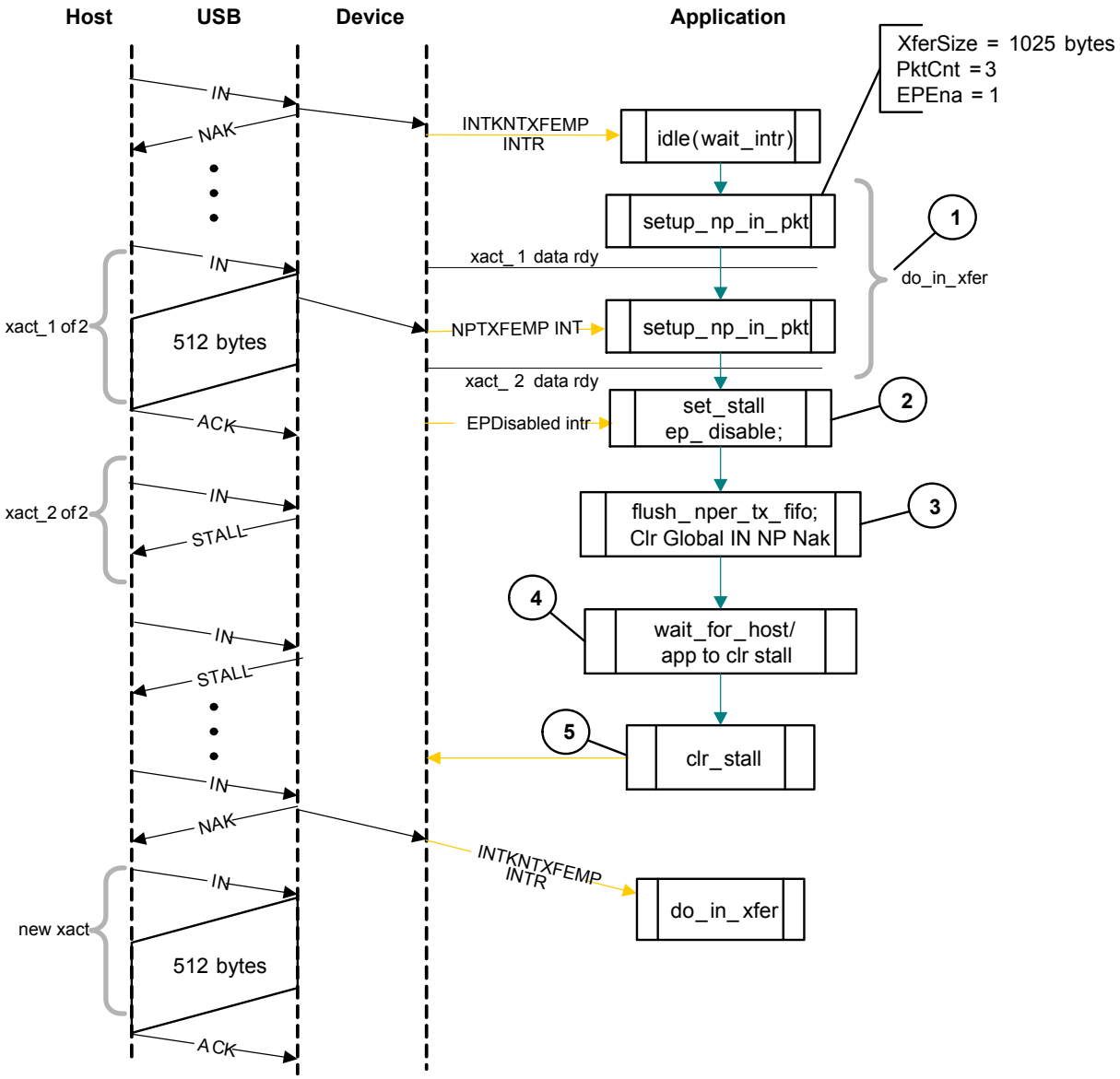


Figure 38.24. USBTRDTIM Max Timing Case ERROR Wrong Image

38.4.4.2.3.10 Handling Babble Conditions

If receives a packet that is larger than the maximum packet size for that endpoint, the core stops writing data to the Rx buffer and waits for the end of packet (EOP). When the core detects the EOP, it flushes the packet in the Rx buffer and does not send any response to the host.

If the core continues to receive data at the EOF2 (the end of frame 2, which is very close to SOF), the core generates an early_suspend interrupt (USB_GINTSTS.ERLYSUSP). On receiving this interrupt, the application must check the erratic_error status bit (USB_DSTS.ERRTICERR). If this bit is set, the application must take it as a long babble and perform a soft reset.

38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode

This section describes a regular non-periodic IN data transfer when transmit thresholding is not enabled.

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. For packet writes in Slave mode, see: 38.4.4.2.3.1 Packet Write in Slave Mode.

Application Requirements

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer is part of a single buffer, and must program the size of that buffer and its start address (in DMA mode) to the endpoint-specific registers.
2. For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   - To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
   - Transfer size[epnum] = n * mps[epnum] + sp

   (where n is an integer >= 0, and 0 <= sp < mps[epnum])
   - If (sp > 0), then packet count[epnum] = n + 1. Otherwise, packet count[epnum] = n

   a. To transmit a single zero-length data packet:
      - Transfer size[epnum] = 0
      - Packet count[epnum] = 1

   b. To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer in two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.

   c. First transfer: transfer size[epnum] = n * mps[epnum]; packet count = n;

   d. Second transfer: transfer size[epnum] = 0; packet count = 1;
3. In DMA mode, the core fetches an IN data packet from the memory, always starting at a DWORD boundary. If the maximum packet size of the IN endpoint is not a multiple of 4, the application must arrange the data in the memory with pads inserted at the end of a maximum-packet-size packet so that a new packet always starts on a DWORD boundary.
4. Once an endpoint is enabled for data transfers, the core updates the Transfer Size register. At the end of IN transfer, which ended with a Endpoint Disabled interrupt, the application must read the Transfer Size register to determine how much data posted in the transmit FIFO was already sent on the USB.
5. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   - Data transmitted on USB = (application-programmed initial packet count – Core updated final packet count) * mps[epnum]
   - Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. In Slave mode, the application must also write the required data to the transmit FIFO for the endpoint. In DMA mode, the core fetches the data from memory according to the application setting for the endpoint.
3. Every time a packet is written into the transmit FIFO, either by the core's internal DMA (in DMA mode) or the application (in Slave Mode), the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory (DMA/Application), until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the "number of packets in FIFO" count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data is written to the transmit FIFO, the core reads it out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a TIMEOUT.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the Packet Count field.
6. If there is no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates a IN Tkn Rcvd When FIFO Empty Interrupt for the endpoint, provided the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. For Control IN endpoint, if there is a TIMEOUT condition, the USB_DIEPx_INT.TIMEOUT interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete interrupt for the endpoint is generated and the endpoint enable is cleared.

Application Programming Sequence

1. Program the USB_DIEPx_TSIZ register with the transfer size and corresponding packet count. In DMA mode, also program the USB_DIEPx_DMAADDR register.

2. Program the USB_DIEPx_CTL register with the endpoint characteristics and set the CNAK and Endpoint Enable bits.
3. In slave mode when transmitting non-zero length data packet, the application must poll the USB_DIEPx_TXFSTS register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use USB_DIEPx_INT.TXFEMP before writing the data.

38.4.4.2.3.12 Examples

Slave Mode Bulk IN Transaction

These notes refer to Figure 38.25 Slave Mode Bulk IN Transaction on page 1524.

1. The host attempts to read data (IN token) from an endpoint.
2. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO.
3. To indicate to the application that there was no data to send, the core generates a USB_DIEPx_INT.INTKNTXFEMP (IN Token Received When TxFIFO Empty) interrupt.
4. When data is ready, the application sets up the USB_DIEPx_TSIZ register with the Transfer Size and Packet Count fields.
5. The application writes one maximum packet size or less of data to the Non-periodic TxFIFO.
6. The host reattempts the IN token.
7. Because data is now ready in the FIFO, the core now responds with the data and the host ACKs it.
8. Because the XFERSIZE is now zero, the intended transfer is complete. The device core generates a USB_DIEPx_INT.XFER-COMPL interrupt.
9. The application processes the interrupt and uses the setting of the USB_DIEPx_INT.XFERCOMPL interrupt bit to determine that the intended transfer is complete.

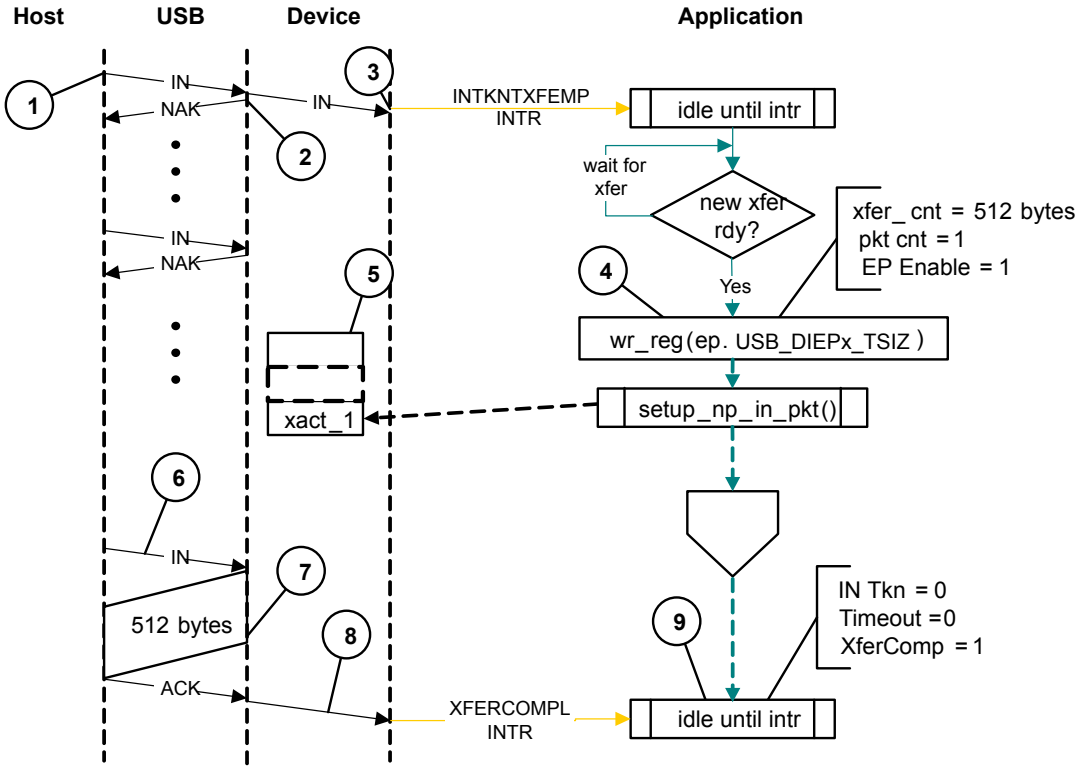


Figure 38.25. Slave Mode Bulk IN Transaction

Slave Mode Bulk IN Transfer (Pipelined Transaction)

These notes refer to Figure 38.26 Slave Mode Bulk IN Transfer (Pipelined Transaction) on page 1525

1. The host attempts to read data (IN token) from an endpoint.
2. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO.
3. To indicate that there was no data to send, the core generates an USB_DIEPx_INT.INTKNTXFEMP (In Token Received When TxFIFO Empty) interrupt.
4. When data is ready, the application sets up the USB_DIEPx_TSIZ register with the transfer size and packet count.
5. The application writes one maximum packet size or less of data to the Non-periodic TxFIFO.
6. The host reattempts the IN token.
7. Because data is now ready in the FIFO, the core responds with the data, and the host ACKs it.
8. When the TxFIFO level falls below the halfway mark, the core generates a USB_GINTSTS.NPTXFEMP (NonPeriodic TxFIFO Empty) interrupt. This triggers the application to start writing additional data packets to the FIFO.
9. A data packet for the second transaction is ready in the TxFIFO.

10. A data packet for third transaction is ready in the TxFIFO while the data for the second packet is being sent on the bus.
11. The second data packet is sent to the host.
12. The last short packet is sent to the host.
13. Because the last packet is sent and XFERSIZE is now zero, the intended transfer is complete. The core generates a USB_DIEPx_INT.XFERCOMPL interrupt.
14. The application processes the interrupt and uses the setting of the USB_DIEPx_INT.XFERCOMPL interrupt bit to determine that the intended transfer is complete

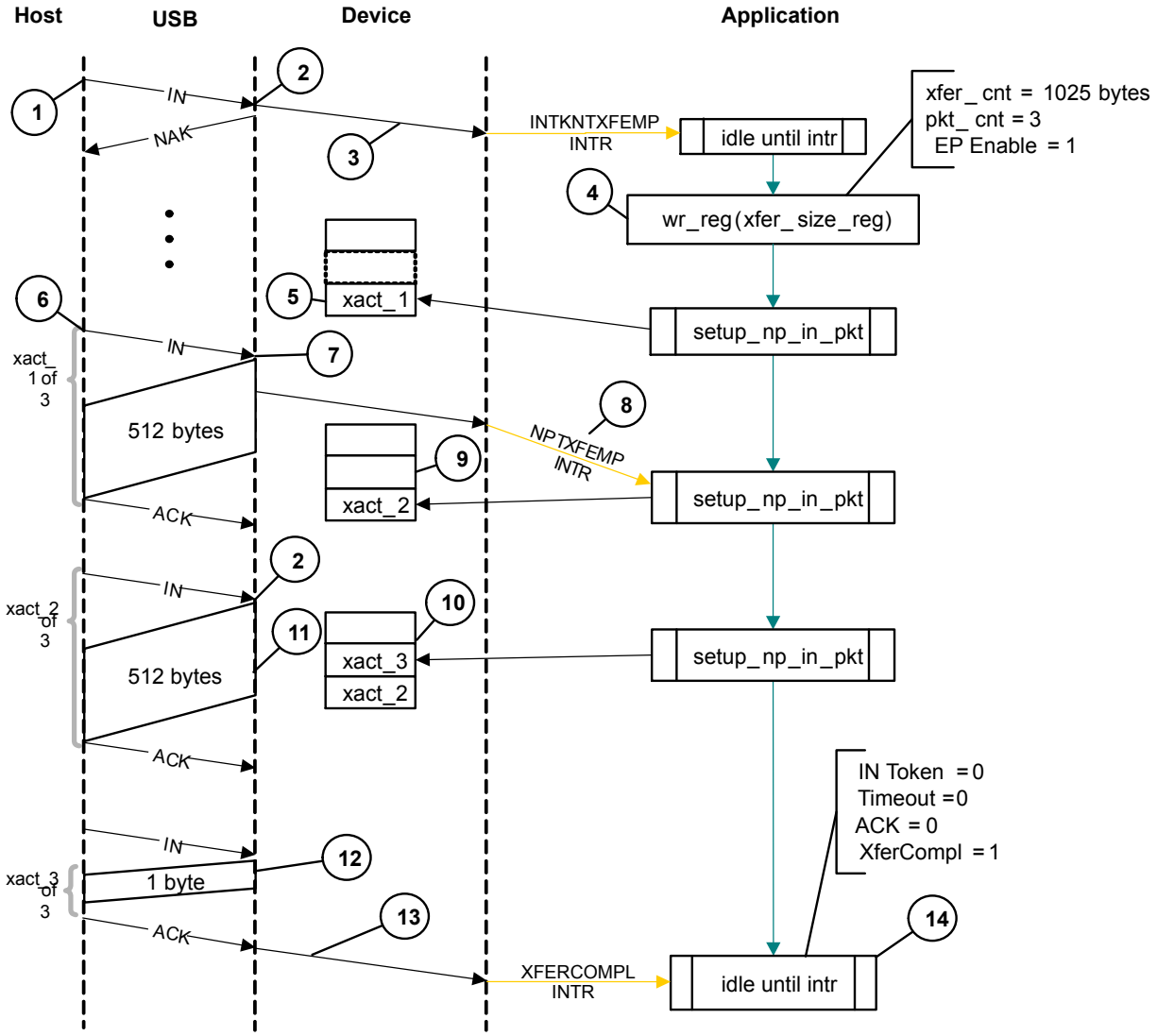


Figure 38.26. Slave Mode Bulk IN Transfer (Pipelined Transaction)

Slave Mode Bulk IN Two-Endpoint Transfer

These notes refer to Figure 38.27 Slave Mode Bulk IN Two-Endpoint Transfer on page 1527

1. The host attempts to read data (IN token) from endpoint 1.
2. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO for endpoint 1, and generates a USB_DIEP1_INT.INTKNTXFEMP (In Token Received When TxFIFO Empty) interrupt.
3. The application processes the interrupt and initializes USB_DIEP1_TSIZ register with the Transfer Size and Packet Count fields. The application starts writing the transaction data to the transmit FIFO.
4. The application writes one maximum packet size or less of data for endpoint 1 to the Non-periodic TxFIFO.
5. Meanwhile, the host attempts to read data (IN token) from endpoint 2.
6. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO for endpoint 2, and the core generates a USB_DIEP2_INT.INTKNTXFEMP (In Token Received When TxFIFO Empty) interrupt.
7. Because the application has completed writing the packet for endpoint 1, it initializes the USB_DIEP2_TSIZ register with the Transfer Size and Packet Count fields. The application starts writing the transaction data into the transmit FIFO for endpoint 2.

8. The host repeats its attempt to read data (IN token) from endpoint 1.
9. Because data is now ready in the TxFIFO, the core returns the data, which the host ACKs.
10. Meanwhile, the application has initialized the data for the next two packets in the TxFIFO (ep2.xact1 and ep1.xact2, in order).
11. The host repeats its attempt to read data (IN token) from endpoint 2.
12. Because endpoint 2's data is ready, the core responds with the data (ep2.xact_1), which the host ACKs.
13. Meanwhile, the application has initialized the data for the next two packets in the TxFIFO (ep2.xact2 and ep1.xact3, in order). The application has finished initializing data for the two endpoints involved in this scenario.
14. The host repeats its attempt to read data (IN token) from endpoint 1.
15. Because data is now ready in the FIFO, the core responds with the data, which the host ACKs.
16. The host repeats its attempt to read data (IN token) from endpoint 2.
17. With data now ready in the FIFO, the core responds with the data, which the host ACKs.
18. With the last packet for endpoint 2 sent and its XFERSIZE now zero, the intended transfer is complete. The core generates a USB_DIEP2_INT.XFERCOMPL interrupt for this endpoint.
19. The application processes the interrupt and uses the setting of the USB_DIEP2_INT.XFERCOMPL interrupt bit to determine that the intended transfer on endpoint 2 is complete.
20. The host repeats its attempt to read data (IN token) from endpoint 1 (last transaction).
21. With data now ready in the FIFO, the core responds with the data, which the host ACKs.
22. Because the last endpoint one packet has been sent and XFERSIZE is now zero, the intended transfer is complete. The core generates a USB_DIEP1_INT.XFERCOMPL interrupt for this endpoint.
23. The application processes the interrupt and uses the setting of the USB_DIEP1_INT.XFERCOMPL interrupt bit to determine that the intended transfer on endpoint 1 is complete.

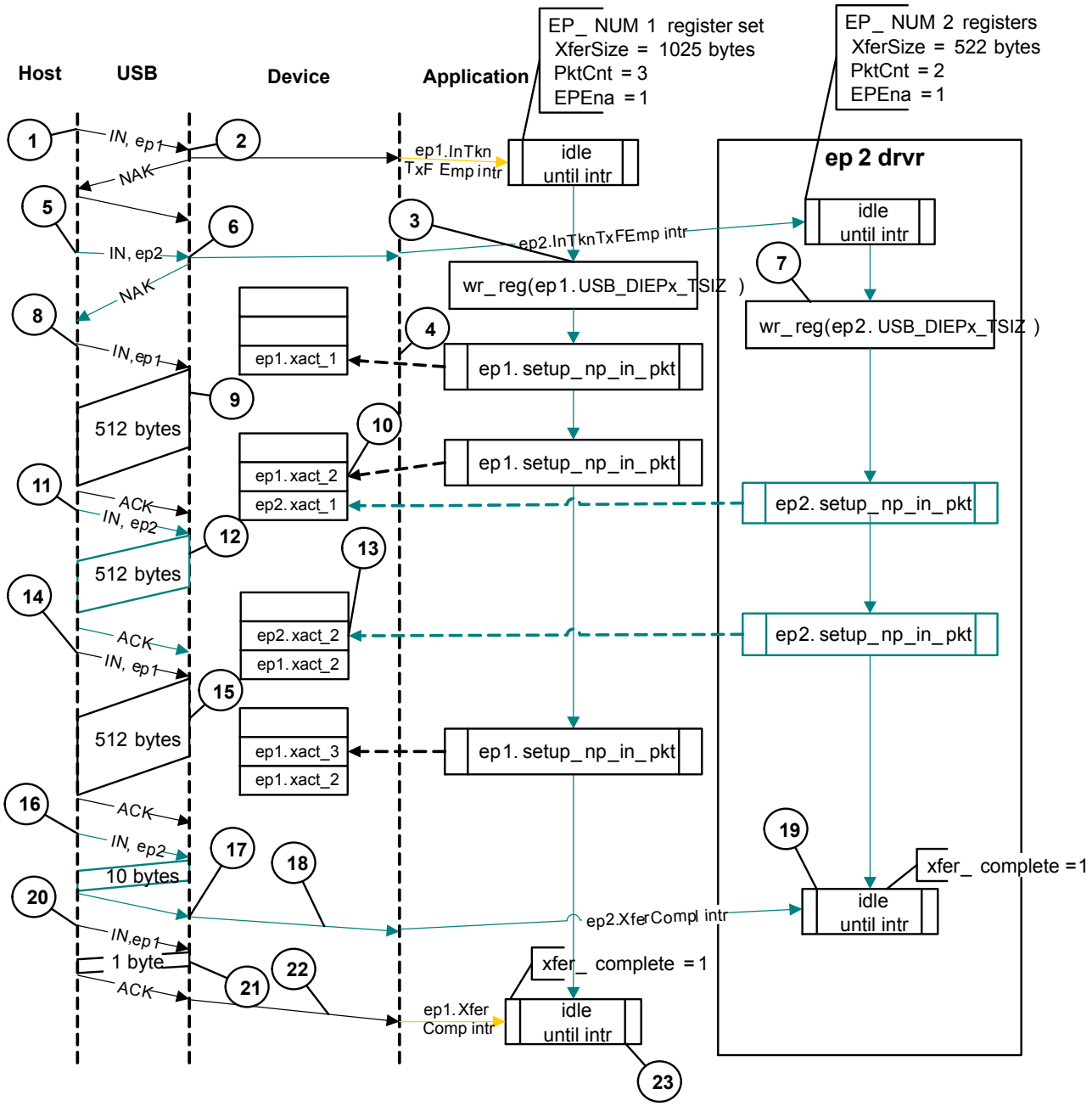


Figure 38.27. Slave Mode Bulk IN Two-Endpoint Transfer

38.4.4.2.3.13 Generic Non-Periodic IN Data Transfers (Bulk and Control) With Thresholding in DMA and Slave Modes

This section describes a regular non-periodic IN data transfer when transmit thresholding is enabled.

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. See 38.4.4.2.2.4 Packet Read From FIFO in Slave Mode.

Application Requirements:

Application requirements are the same as those for DMA mode with no thresholding. See 38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode.

Internal Data Flow:

1. The application must set the transfer size and packet count fields in the Endpoint Specific registers, and enable the endpoint to transmit the data.
2. The core fetches an AHB threshold amount of data for one endpoint before switching to the next in a round-robin fashion with equal fairness. The priority is given to periodic endpoints. non-periodic endpoint data is fetched only if data for the periodic endpoints has been fetched or if the currently active token on the USB is a non-periodic one.The core does not switch, and continues to fetch until the complete packet when it finds that the currently active IN token on the USB is for that particular endpoint and the FIFO does not have the complete packet.
3. In response to an IN token on the USB, the core starts transmitting data, if the MAC finds at least a MAC threshold amount of data in the FIFO for that particular endpoint.
4. With each AHB threshold amount of data written into the FIFO, the transfer size for that endpoint is decremented by the AHB threshold size, except for the last packet. For the last packet, the transfer size is not decremented by the core. After writing the first threshold amount of data into the FIFO, the “number of packets in FIFO” count is incremented. For zero-length packets, a separate flag is set for that endpoint FIFO, without any data in the FIFO. This count is internally maintained by the core and is decremented when a full packet has been read out of the FIFO.
5. If the MAC sees an underrun case, where there is not enough data in the FIFO, the core corrupts the data (inverts the CRC) on the USB. The core internally flushes the FIFO, rewinding the DMA pointers and re-fetching the packets. Additionally, the USB_DIEPx_INT.TXFIFOUNDRN bit is set as an indication to the application.
6. For every non-ISO data IN packet transmitted, with an ACK handshake, the packet count for the endpoint is decremented by 1, until the packet count becomes zero. The packet count is not decremented on a TIMEOUT or underrun condition.
7. If the zero length flag in the FIFO is set (internally set by the core, when the application enables and endpoint for zero length), then core sends out zero length packet for the IN token and decrements the packet count field.
8. If there is no data (or partial threshold data) in the FIFO for a received IN token, and the packet count field for that endpoint is 0, the core generates a IN Tkn Rcvd When FIFO Empty Interrupt for the endpoint. The core responds with a NAK handshake for the non-ISO endpoints on the USB.
9. If the core does not receive a handshake after sending the packet (TIMEOUT), the core internally flush the FIFO, rewind the DMA pointers and re-fetch the packet(s).
10. When the packet count is zero, the transfer complete interrupt for the endpoint is generated, and then the endpoint enable and transfer size are both cleared by the core.

Application Programming Sequence:

1. Program the USB_DIEPx_TSIZ register, for the transfer size and the corresponding packet count and the USB_DIEPx_DMAADDR register.
2. Program the USB_DIEPx_CTL register, with the Endpoint Characteristics and set the CNAK and the Endpoint Enable bit.Also specify the Tx FIFO number in the USB_DIEPx_CTL.TXFNUM field.
3. Assertion of USB_DIEPx_INT.XFERCOMPL interrupt marks the successful completion of the non-periodic IN transfer. Read to USB_DIEPx_TSIZ register must indicate, a transfer size = 0 and packet count = 0, indicating all the data is transmitted on the USB.

Example Transfer Data Flow for Bulk IN DMA Mode with Thresholding

These notes refer to Figure 38.28 Bulk IN DMA Mode With Thresholding on page 1530

1. The host attempts to read data (IN token) from an endpoint
2. On receiving the IN token on the USB bus, the core sends back a NAK handshake because no data is available in the Transmit FIFO
3. To indicate to the application that there was no data to send, the core generates a USB_DIEPx_INT.INTKNTXFEMP (IN Token Received When TxFifo Empty) interrupt.
4. When data is ready, the application sets up the USB_DIEPx_TSIZ register with the TransferSize and Packet Count fields and enables the endpoint.
5. On seeing the endpoint enabled, the internal DMA engine start fetching the first threshold amount of data.
6. DMA engine fetching the second threshold.

7. The host re-attempts the IN token, and core start sending the data because it sees at least threshold amount of data in the FIFO.
8. The DMA engine starts fetching the third threshold after it sees threshold amount of space.
9. The MAC sees an underrun condition because of FIFO being empty in the middle of the packet, stops the packet and corrupt the CRC.
10. DMA engine starts to fetch the last threshold for that packet but it is late and eventually results in an underrun.
11. No handshake response from the host, because the packet was corrupted. Core rewinds the pointers and flush the FIFO.
12. The core starts to re-fetch the packet, staring with the first threshold.
13. The host re-attempts the IN token, and the core starts sending the data, since the core detects at least the threshold amount of data in the FIFO.
14. The core fetches the last threshold in time.
15. The core receives the handshake from host and Because the XFERSIZE is now zero, the intended transfer is complete. Device core generates a USB_DIEPx_INT.XFERCOMPL interrupt.
16. The application processes the interrupt and uses the setting of USB_DIEPx_INT.XFERCOMPL interrupt bit to determine that the intended transfer is complete.

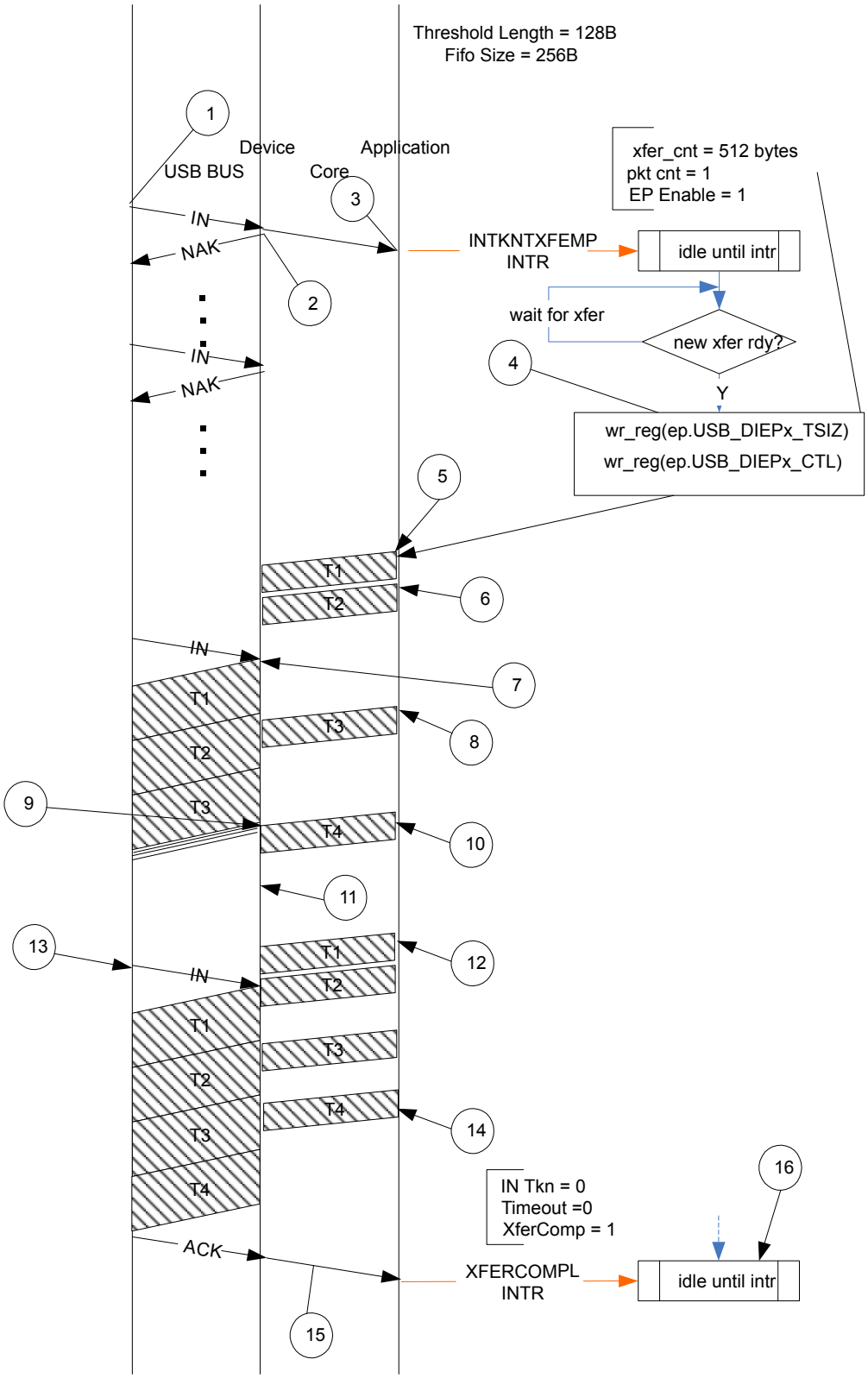


Figure 38.28. Bulk IN DMA Mode With Thresholding

38.4.4.2.3.14 Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding

This section describes a typical periodic IN data transfer when thresholding is not enabled.

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. For packet writes in Slave mode, see: 38.4.4.2.3.1 Packet Write in Slave Mode.

Application Requirements

1. Application requirements 1, 2, 3, and 4 of 38.4.4.2.3.11 Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode also apply to periodic IN data transfers, except for a slight modification of Requirement 2.
   - The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met.
     - transfer size[epnum] = n * mps[epnum] + sp(where n is an integer 0, and 0 >= sp < mps[epnum])
     - If (sp > 0), packet count[epnum] = n + 1Otherwise, packet count[epnum] = n;
     - mc[epnum] = packet count[epnum]
   - The application cannot transmit a zero-length data packet at the end of transfer. It can transmit a single zero-length data packet by it self. To transmit a single zero-length data packet,
   - transfer size[epnum] = 0
     - packet count[epnum] = 1
     - mc[epnum] = packet count[epnum]
2. The application can only schedule data transfers 1 frame at a time.
   - (USB_DIEPx_TSIZ.MC – 1) * USB_DIEPx_CTL.MPS =< USB_DIEPx_TSIZ.XFERSIZE =< USB_DIEPx_TSIZ.MC * USB_DIEPx_CTL.MPS
   - USB_DIEPx_TSIZ.PKTCNT = USB_DIEPx_TSIZ.MC
   - If USB_DIEPx_TSIZ.XFERSIZE < USB_DIEPx_TSIZ.MC * USB_DIEPx_CTL.MPS, the last data packet of the transfer is a short packet.
3. This step is not applicable for isochronous data transfers, only for interrupt transfers.

   The application can schedule data transfers for multiple frames, only if multiples of max packet sizes (up to 3 packets), must be transmitted every frame. This is can be done, only when the core is operating in DMA mode. This is not a recommended mode though.
   - ((n*USB_DIEPx_TSIZ.MC) - 1)*USB_DIEPx_CTL.MPS <= USB_DIEPx_TSIZ.XFERSIZE <= n*USB_DIEPx_TSIZ.MC*USB_DIEPx_CTL.MPS
   - USB_DIEPx_TSIZ.PKTCNT = n*USB_DIEPx_TSIZ.MC
   - n is the number of frames for which the data transfers are scheduled

   Data Transmitted per frame in this case would be USB_DIEPx_TSIZ.MC*USB_DIEPx_CTL.MPS, in all the frames except the last one. In the frame “n”, the data transmitted would be (USB_DIEPx_TSIZ.XFERSIZE - (n-1)*USB_DIEPx_TSIZ.MC*USB_DIEPx_CTL.MPS)
4. For Periodic IN endpoints, the data must always be prefetched 1 frame ahead for transmission in the next frame. This can be done, by enabling the Periodic IN endpoint 1 frame ahead of the frame in which the data transfer is scheduled.
5. The complete data to be transmitted in the frame must be written into the transmit FIFO (either by the application or the DMA), before the Periodic IN token is received. Even when 1 DWORD of the data to be transmitted per frame is missing in the transmit FIFO when the Periodic IN token is received, the core behaves as when the FIFO was empty. When the transmit FIFO is empty,
6. A zero data length packet would be transmitted on the USB for ISO IN endpoints
   - A NAK handshake would be transmitted on the USB for INTR IN endpoints
7. For a High Bandwidth IN endpoint with three packets in a frame, the application can program the endpoint FIFO size to be 2*max_pkt_size and have the third packet load in after the first packet has been transmitted on the USB.

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. In Slave mode, the application must also write the required data to the associated transmit FIFO for the endpoint. In DMA mode, the core fetches the data for the endpoint from memory, according to the application setting.
3. Every time either the core’s internal DMA (in DMA mode) or the application (in Slave mode) writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from DMA or application memory until the transfer size for the endpoint becomes 0.

4. When an IN token is received for an periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet) for the frame is not present in the FIFO, then the core generates an IN Token Received When TxFIFO Empty Interrupt for the endpoint.
   - A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   - A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
   - For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   - For interrupt endpoints, when an ACK handshake is transmitted
   - When the transfer size and packet count are both 0, the Transfer Completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by USB_DCFG.PERFRINT), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates a USB_GINTSTS.INCOMPISOIN interrupt.

Application Programming Sequence (Transfer Per Frame)

1. Program the USB_DIEPx_TSIZ register. In DMA mode, also program the USB_DIEPx_DMAADDR register.
2. Program the USB_DIEPx_CTL register with the endpoint characteristics and set the CNAK and Endpoint Enable bits.
3. In Slave mode, write the data to be transmitted in the next frame to the transmit FIFO.
4. Asserting the USB_DIEPx_INT.INTKNTXFEMP (In Token Received When TxFifo Empty) interrupt indicates that either the DMA or application has not yet written all data to be transmitted to the transmit FIFO.
5. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
   - If the isochronous endpoint is already enabled when this interrupt is detected, see 38.4.4.2.3.6 Incomplete Isochronous IN Data Transfers for more details.
6. The core handles timeouts internally on interrupt IN endpoints programmed as periodic endpoints without application intervention. The application, thus, never detects a USB_DIEPx_INT.TIMEOUT interrupt for periodic interrupt IN endpoints.
7. Asserting the USB_DIEPx_INT.XFERCOMPL interrupt with no USB_DIEPx_INT.INTKNTXFEMP (In Token Received When TxFifo Empty) interrupt indicates the successful completion of an isochronous IN transfer. A read to the USB_DIEPx_TSIZ register must indicate transfer size = 0 and packet count = 0, indicating all data is transmitted on the USB.
8. Asserting the USB_DIEPx_INT.XFERCOMPL interrupt, with or without the USB_DIEPx_INT.INTKNTXFEMP (In Token Received When TxFifo Empty) interrupt, indicates the successful completion of an interrupt IN transfer. A read to the USB_DIEPx_TSIZ register must indicate transfer size = 0 and packet count = 0, indicating all data is transmitted on the USB.
9. Asserting the USB_GINTSTS.INCOMPISOIN (Incomplete Isochronous IN Transfer) interrupt with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.
10. For isochronous IN endpoints, see 38.4.4.2.3.6 Incomplete Isochronous IN Data Transfers, for more details.

38.4.4.2.3.15 Generic Periodic IN Data Transfers Without Thresholding Using the Periodic Transfer Interrupt Feature

This section describes a typical Periodic IN (ISOC / INTR) data transfer with the Periodic Transfer Interrupt feature, when Thresholding is not enabled.

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer is part of a single buffer, and must program the size of that buffer and its start address (in DMA mode) to the endpoint-specific registers.
2. For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   a. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
      - Transfer size[epnum] = n * mps[epnum] + sp

      (where n is an integer > 0, and 0 < sp < mps[epnum]. A higher value of n reduces the periodicity of the USB_DOEPx_INT.XFERCOMPL interrupt)
      - If (sp > 0), then packet count[epnum] = n + 1. Otherwise, packet count[epnum] = n

   b. To transmit a single zero-length data packet:
      - Transfer size[epnum] = 0
      - Packet count[epnum] = 1

   c. To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer in two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
      - First transfer: transfer size[epnum] = n * mps[epnum]; packet count = n;
      - Second transfer: transfer size[epnum] = 0; packet count = 1;

   d. The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met.
      - transfer size[epnum] = n * mps[epnum] + sp (where n is an integer > 0, and 0 < sp < mps[epnum])
      - If (sp > 0), packet count[epnum] = n + 1 Otherwise, packet count[epnum] = n;
      - mc[epnum] = number of packets to be sent out in a frame.

   e. The application cannot transmit a zero-length data packet at the end of transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet,
      - transfer size[epnum] = 0
      - packet count[epnum] = 1
      - mc[epnum] = packet count[epnum]
3. In DMA mode, the core fetches an IN data packet from the memory, always starting at a DWORD boundary. If the maximum packet size of the IN endpoint is not a multiple of 4, the application must arrange the data in the memory with pads inserted at the end of a maximum-packet-size packet so that a new packet always starts on a DWORD boundary.
4. Once an endpoint is enabled for data transfers, the core updates the Transfer Size register. At the end of IN transfer, which ended with a Endpoint Disabled interrupt, the application must read the Transfer Size register to determine how much data posted in the transmit FIFO was already sent on the USB.
   - Data fetched into transmit FIFO = Application-programmed initial transfer size - core-updated final transfer size
   - Data transmitted on USB = (application-programmed initial packet count - Core updated final packet count) * mps[epnum]
   - Data yet to be transmitted on USB = (Application-programmed initial transfer size - data transmitted on USB)
5. The application can schedule data transfers for multiple frames, only if multiples of max packet sizes (up to 3 packets), must be transmitted every frame. This is can be done, only when the core is operating in DMA mode.
   - ((n*USB_DIEPx_TSIZ.MC) - 1)*USB_DIEPx_CTL.MPS <= USB_DIEPx_TSIZ.XFERSIZE <= n*USB_DIEPx_TSIZ.MC*USB_DIEPx_CTL.MPS
   - USB_DIEPx_TSIZ.PKTCNT = n*USB_DIEPx_TSIZ.MC
   - n is the number of frames for which the data transfers are scheduled. Data Transmitted per frame in this case is USB_DIEPx_TSIZ.MC*USB_DIEPx_CTL.MPS in all frames except the last one. In frame n, the data transmitted is (USB_DIEPx_TSIZ.XFERSIZE – (n – 1) * USB_DIEPx_TSIZ.MC * USB_DIEPx_CTL.MPS)
6. For Periodic IN endpoints, the data must always be prefetched 1 frame ahead for transmission in the next frame. This can be done, by enabling the Periodic IN endpoint 1 frame ahead of the frame in which the data transfer is scheduled.
7. The complete data to be transmitted in the frame must be written into the transmit FIFO, before the Periodic IN token is received. Even when 1 DWORD of the data to be transmitted per frame is missing in the transmit FIFO when the Periodic IN token is received, the core behaves as when the FIFO was empty. When the transmit FIFO is empty,
   - A zero data length packet would be transmitted on the USB for ISOC IN endpoints
   - A NAK handshake would be transmitted on the USB for INTR IN endpoints
   - USB_DIEPx_TSIZ.PKTCNT is not decremented in this case.

8. For a High Bandwidth IN endpoint with three packets in a frame, the application can program the endpoint FIFO size to be 2 * max_pkt_size and have the third packet load in after the first packet has been transmitted on the USB.

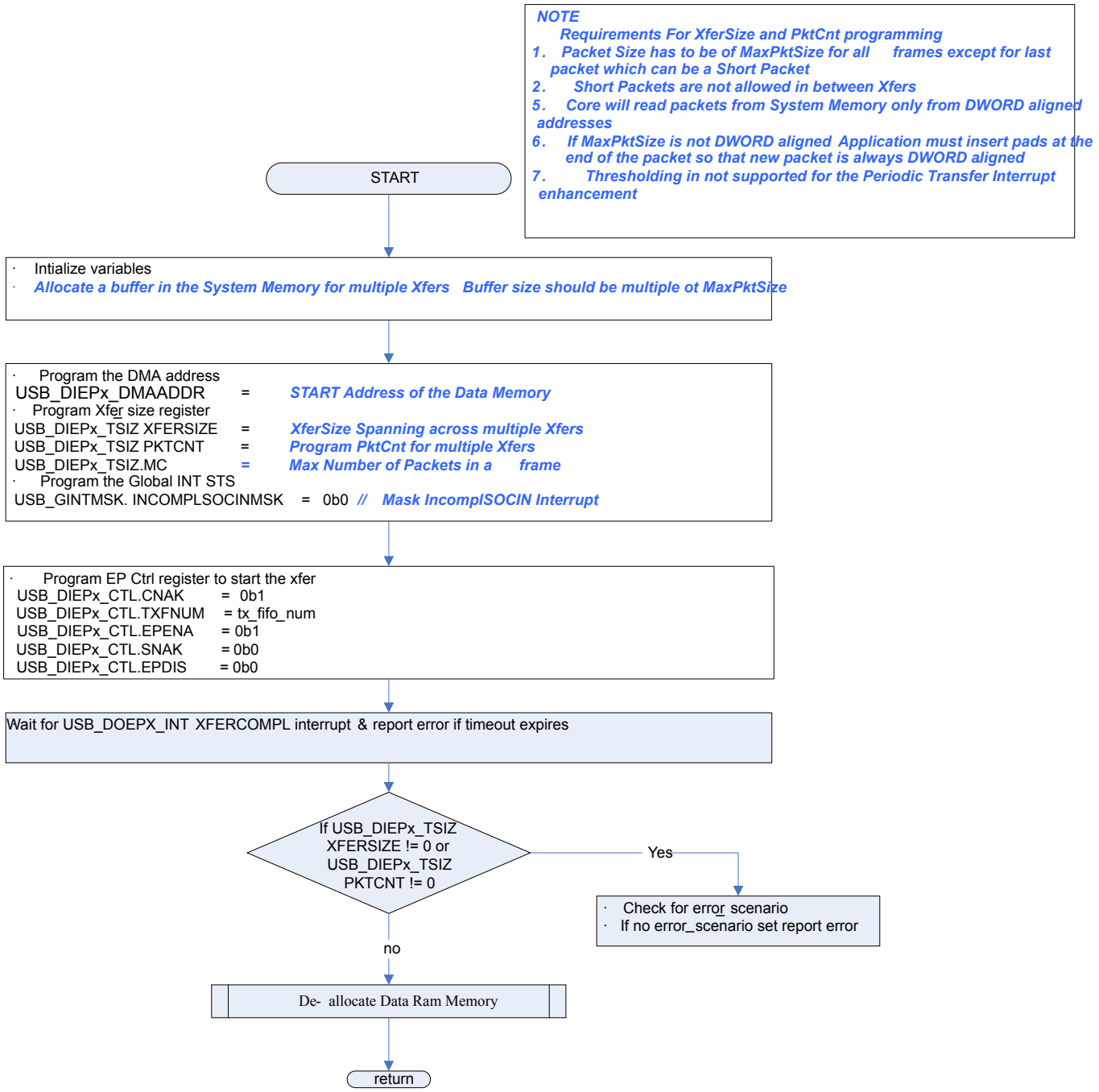


Figure 38.29. Periodic in Application Flow for Periodic Transfer Interrupt Feature

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
   - The application must enable the USB_DCTL.IGNRFRMNUM
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame will be ignored by the core.
   - Subsequently the core updates the Even / Odd bit on its own
3. Every time either the core's internal DMA writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from DMA or application memory until the transfer size for the endpoint becomes 0.

4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet) for the frame is not present in the FIFO, then the core generates an IN Token Received When TxFifo Empty Interrupt for the endpoint.
   - A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   - A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. If an IN token comes for an endpoint on the bus, and if the corresponding TxFIFO for that endpoint has at least 1 packet available, and if the USB_DIEPx_CTL.NAK bit is not set, and if the internally maintained even/odd bit match with the bit 0 of the current frame number, then the core will send this data out on the USB. The core will also decrement the packet count. Core also toggles the MultCount in USB_DIEPx_CTL register and based on the value of MultCount the next PID value is sent.
   - If the IN token results in a timeout (core did not receive the handshake or handshake error), core rewind the FIFO pointers. Core does not decrement packet count. It does not toggle PID. USB_DIEPx_INT.TIMEOUT interrupt will be set which the application could check.
   - At the end of periodic frame interval (Based on the value programmed in the USB_DCFG.PERFRINT register, core will internally set the even/odd internal bit to match the next frame.
6. The packet count for the endpoint is decremented by 1 under the following conditions:
   - For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   - For interrupt endpoints, when an ACK handshake is transmitted
7. The data PID of the transmitted data packet is based on the value of USB_DIEPx_TSIZ.MC programmed by the application. In case the USB_DIEPx_TSIZ.MC value is set to 3 then, for a particular frame the core expects to receive 3 Isochronous IN token for the respective endpoint. The data PIDs transmitted will be D2 followed by D1 and D0 respectively for the tokens.
   - If any of the tokens responded with a zero-length packet due to non-availability of data in the TxFIFO, the packet is sent in the next frame with the pending data PID. For example, in a frame, the first received token is responded to with data and data PID value D2. If the second token is responded to with a zero-length packet, the host is expected not to send any more tokens for the respective endpoint in the current frame. When a token arrives in the next frame it will be responded to with the pending data PID value of D1.
   - Similarly the second token of the current frame gets responded with D0 PID. The host is expected to send only two tokens for this frame as the first token got responded with D1 PID.
8. When the transfer size and packet count are both 0, the Transfer Completed interrupt for the endpoint is generated and the endpoint enable is cleared.
9. The USB_GINTSTS.INCOMPISOIN will be masked by the application hence at the Periodic Frame interval (controlled by USB_DCFG.PERFRINT), even though the core finds non-empty any of the isochronous IN endpoint FIFOs, USB_GINTSTS.INCOMPISOIN interrupt will not be generated.

38.4.4.2.3.16 Generic Periodic (Interrupt and ISO) in Data Transfers With Thresholding

This section describes a typical Periodic IN data transfer when thresholding is enabled.

To initialize the core after power-on reset, the application must follow the sequence in 38.4.1 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in 38.4.4.1 Endpoint Initialization. For packet writes in Slave mode, see: 38.4.4.2.3.1 Packet Write in Slave Mode.

Application Requirements

Application requirements are the same as that for DMA mode with no thresholding.

Internal Data Flow

1. The application must set the transfer size and packet count fields in the Endpoint Specific registers, and enable the endpoint to transmit the data.
2. The core fetches an AHB threshold amount of data for one endpoint before switching to the next in a round robin fashion with equal fairness. The priority is given to periodic endpoints. The core does not switch, and continues to fetch until the complete packet, if it finds that the currently active IN token on the USB side is for that particular endpoint and the FIFO does not have the complete packet.
3. In response to an IN token on the USB, the core starts transmitting, if the MAC finds at least one MAC threshold amount of data in the FIFO for that particular endpoint.
4. After a full packet has been written into the FIFO, the transfer size for that endpoint is decremented by the packet size (or less). After writing the first AHB threshold amount of data into the FIFO, the “number of packets in FIFO” count is incremented. For zero length packets, a separate flag is set in the FIFO, without any data in the FIFO. This count is internally maintained by the core and is decremented when a full packet has been read out of the FIFO.
5. If the MAC sees an underrun case, where there is not enough data in the FIFO, the core corrupts the data (inverts the CRC) on the USB. For isochronous endpoints, Underrun interrupt (USB_DIEPx_INT.TxFifoUndrn) is generated by the core for this endpoint. For interrupt endpoints, the core flushes the FIFO, rewinds the DMA pointers and re-fetches the data.
6. If the core sees a timeout or an underrun condition for an interrupt endpoint, the core flushes the fifo, rewinds the DMA pointers and re-fetches the packet. No interrupt is generated to the application.
7. When an IN token is received for an periodic endpoint, the core transmits the data in the FIFO, if the core sees at least the MAC threshold amount of data. If the MAC threshold amount of data for the packet is not present in the FIFO, the core generates a IN Tkn Rcvd When TxF Empty Interrupt for the endpoint.
   - A zero data length packet would be transmitted on the USB for ISO IN endpoints
   - A NAK handshake would be transmitted on the USB for INTR IN endpoints
8. The packet count for the endpoint is decremented by 1, on the following conditions
   - When a zero or non zero data length for ISO endpoints
   - When an ACK handshake is transmitted for INTR endpoints
9. The packet count is not decremented when the core has to corrupt a packet because of underrun condition.
10. When the transfer size is 0 and the packet count is 0, the transfer complete interrupt for the endpoint is generated and the endpoint enable is cleared.

Application Programming Sequence (Transfer Per Frame)

1. Program the USB_DIEPx_TSIZ register and in addition, in DMA mode program the USB_DIEPx_DMAADDR register.
2. Program the USB_DIEPx_CTL register, with the Endpoint Characteristics and set the CNAK bit and the Endpoint Enable bit.Also specify the Tx FIFO number in the USB_DIEPx_CTL.TXFNUM field.
3. Assertion of USB_DIEPx_INT.INTKNTXFEMP (In Token Received When TxFifo Empty) interrupt indicates that the complete data to be transmitted is not written into the transmit FIFO.
4. If the INTR endpoint is already enabled, when this interrupt is seen, ignore the interrupt. If the IN Endpoint is not enabled, enable the endpoint, so that the data could be transmitted on the next attempt of the IN token.
   - If the ISO endpoint is already enabled, when this interrupt is seen. For ISO IN endpoints, see 38.4.4.2.3.6 Incomplete Isochronous IN Data Transfers for details.
5. TIMEOUT on Interrupt IN endpoints, is handled by the core internally, without any application intervention. The application never sees any interrupt for timeout.
6. Assertion of USB_DIEPx_INT.XFERCOMPL interrupt without any USB_DIEPx_INT.INTKNTXFEMP (In Token Received When TxFifo Empty) interrupt, marks the successful completion of the ISO IN transfer. Read to USB_DIEPx_TSIZ register must indicate, a transfer size = 0 and packet count = 0, indicating all the data is transmitted on the USB.
7. Assertion of USB_DIEPx_INT.XFERCOMPL interrupt with/without any USB_DIEPx_INT.INTKNTXFEMP (In Token Received When TxFifo Empty) interrupt, marks the successful completion of the INTR IN transfer. Read to USB_DIEPx_TSIZ register must indicate, a transfer size = 0 and packet count = 0, indicating all the data is transmitted on the USB.

8. Assertion of USB_GINTSTS.INCOMPISOIN (Incomplete ISO IN Transfer) interrupt without any of the previously mentioned interrupts indicates that at least 1 periodic IN token was not received by the core, in the current frame. For ISO IN endpoints, see 38.4.4.2.3.6 Incomplete Isochronous IN Data Transfers for details.
9. Assertion of USB_DIEPx_INT.TXFIFOUNDRN interrupt indicate that there was an underrun condition for the isochronous transaction in the current frame. The application may choose ignore this interrupt, as this eventually results in a USB_GINTSTS.INCOMPISOIN (Incomplete isochronous IN) interrupt at the end of a periodic frame. The application can also choose to service this interrupt. If they choose to do so, then they can save some time in re-enabling the endpoint for the next frame. In response to this interrupt,
10. Disable the endpoint.
    - Application reads Endpoint USB_DIEPx_TSIZ register to see how much data is transferred.
    - Re-enable the endpoint for the next frame.

Example Transfer Data Flow for Isochronous IN DMA Mode With Thresholding

Figure 38.30 Isochronous IN DMA Mode With Thresholding on page 1538 shows an isochronous In transfer when thresholding is enabled. The FIFO size is assumed to be 512 bytes and the threshold length is 128 bytes.

1. The application enables the isochronous IN endpoint for odd frame.
2. The core starts fetching the first two AHB thresholds.
3. The core starts sending data out in response to the IN token.
4. The core sees an underrun condition, because the third threshold was not fetched in time.
5. The core generates USB_DIEPx_INT.TXFIFOUNDRN interrupt (In this case, application ignores this interrupt).
6. At the End of Periodic Frame Interval, core generates USB_GINTSTS.INCOMPISOIN interrupt.

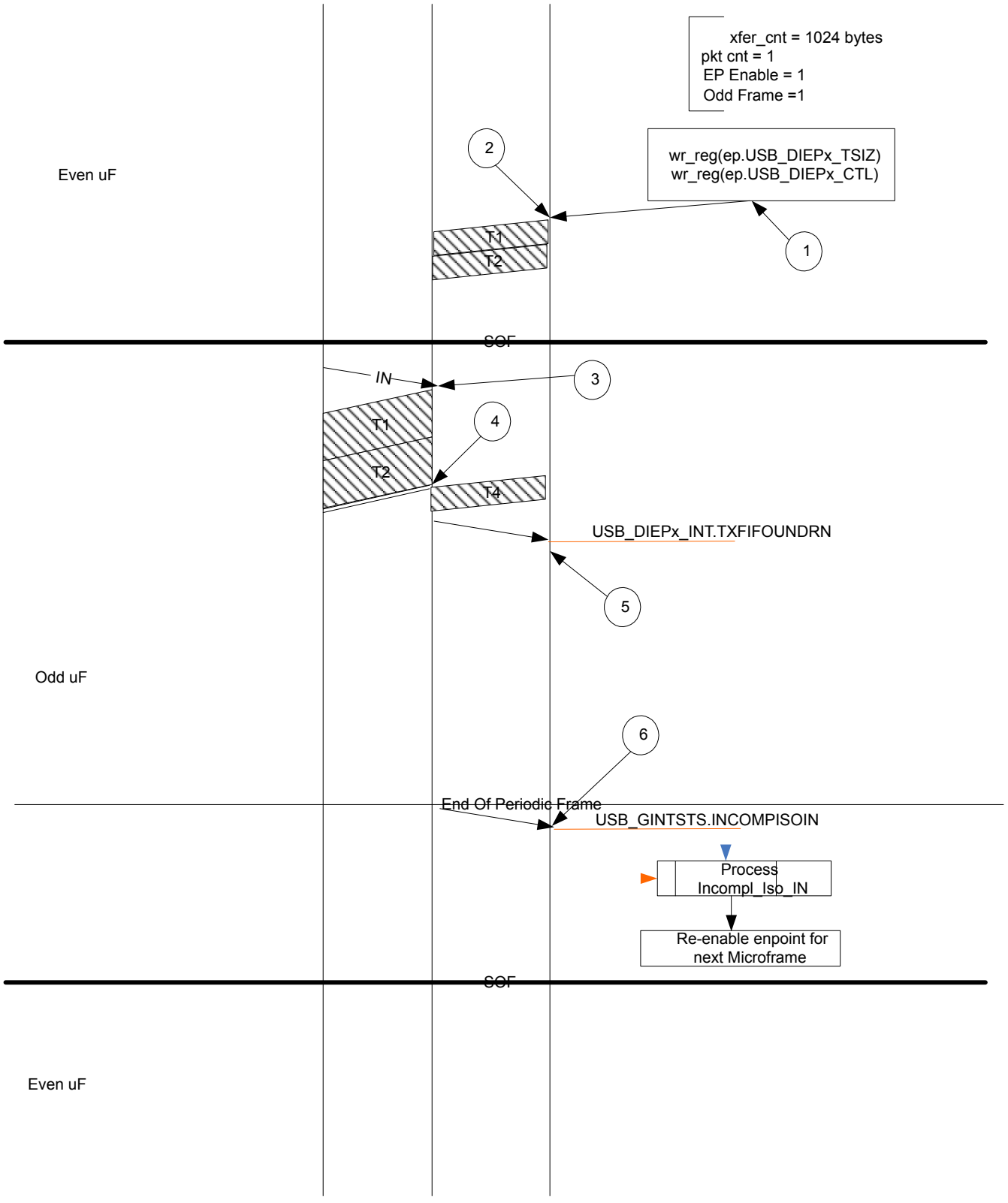


Figure 38.30. Isochronous IN DMA Mode With Thresholding

38.4.5 FIFO RAM Allocation

The follwing sections detail how data FIFO RAM is allocated.

38.4.5.1 Data FIFO RAM Allocation

External RAM must be allocated among different FIFOs in the core before any transactions can start. The application must follow this procedure every time it changes core FIFO RAM allocation.

The application must allocate data RAM per FIFO based on the AHB's operating frequency, the PHY Clock frequency, the available AHB bandwidth, and the performance required on the USB. Based on the above mentioned criteria, the application must provide a table as described below with RAM sizes for each FIFO in each mode.

The core shares a single FIFO RAM between transmit FIFO(s) and receive FIFO.

In DMA mode—The FIFO RAM is also used for storing the some register information.

The Device mode Endpoint DMA address registers (USB_DIEP0DMAADDR, USB_DOEP0DMAADDR, USB_DIEPx_DMAADDR, USB_DOEPx_DMAADDR) and Host mode Channel DMA registers (USB_HCx_DMAADDR) are stored in the FIFO RAM.

- These register information are stored at the end of the FIFO RAM after the space allocated for receive and Transmit FIFO. These register space must also be taken into account when calculating the total FIFO depth of the core as explained in the following sections.

The registers USB_DIEPx_DMAADDR/USB_DOEPx_DMAADDR are maintained in RAM.

The following rules apply while calculating how much RAM space must be allocated to store these registers.

Host Mode:

- Slave mode only: No space needed.
- DMA mode: One location per channel.

Device Mode:

- Slave mode only: No space needed.
- DMA mode: One location per end point direction.

38.4.5.1.1 Device Mode

When allocating data RAM for FIFOs in Device mode keep in mind the factors in the follwing sections.

38.4.5.1.1.1 Tx FIFO Operation Without Thresholding

1. Receive FIFO RAM allocation:
   - RAM for SETUP Packets: 4 * n + 6 locations must be Reserved in the receive FIFO to receive up to n SETUP packets on control endpoints, where n is the number of control endpoints the device core supports. The core does not use these locations, which are Reserved for SETUP packets, to write any other data.
   - One location for Global OUT NAK
   - Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (Largest Packet Size / 4) + 1 must be allotted to receive packets. If a high-bandwidth endpoint is enabled, or multiple isochronous endpoints are enabled, then at least two (Largest Packet Size / 4) + 1 spaces must be allotted to receive back-to-back packets. Typically, two (Largest Packet Size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to AHB, the USB can receive the subsequent packet. If AHB latency is high, you must allocate enough space to receive multiple packets. This is critical to prevent dropping any isochronous packets.
   - Along with each endpoint's last packet, transfer complete status information is also pushed to the FIFO. Typically, one location for each OUT endpoint is recommended.
2. Transmit FIFO RAM Allocation:

   The minimum RAM space required for each IN Endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

   More space allocated in the transmit IN Endpoint FIFO results in a better performance on the USB and can hide latencies on the AHB.

FIFO Name	Data RAM Size
Receive data FIFO	rx_fifo_size. This must include RAM for setup packets, OUT endpoint control information and data OUT packets, as mentioned earlier.
Transmit FIFO 0	tx_fifo_size[0]
Transmit FIFO 1	tx_fifo_size[1]
Transmit FIFO 2	tx_fifo_size[2]
...	...
Transmit FIFO i	tx_fifo_size[i]

With this information, the following registers must be programmed as follows:

1. Receive FIFO Size Register (USB_GRXFSIZ)

   USB_GRXFSIZ.Receive FIFO Depth = rx_fifo_size;
2. Device IN Endpoint Transmit FIFO0 Size Register (USB_GNPTXFSIZ)

   USB_GNPTXFSIZ.non-periodic Transmit FIFO Depth = tx_fifo_size[0];

   USB_GNPTXFSIZ.non-periodic Transmit RAM Start Address = rx_fifo_size;
3. Device IN Endpoint Transmit FIFO#1 Size Register (USB_DIEPTXF1)

   USB_DIEPTXF1. Transmit RAM Start Address = USB_GNPTXFSIZ.FIFO0 Transmit RAM Start Address + tx_fifo_size[0];
4. Device IN Endpoint Transmit FIFO#2 Size Register (USB_DIEPTXF2)

   USB_DIEPTXF2.Transmit RAM Start Address = USB_DIEPTXF1.Transmit RAM Start Address + tx_fifo_size[1];
5. Device IN Endpoint Transmit FIFO#i Size Register (USB_DIEPTXFi)

   USB_DIEPTXFm.Transmit RAM Start Address = USB_DIEPTXFi-1.Transmit RAM Start Address + tx_fifo_size[i-1];
6. The transmit FIFOs and receive FIFO must be flushed after the RAM allocation is done, for the proper functioning of the FIFOs.
   - USB_GRSTCTL.TXFNUM = 0x10
   - USB_GRSTCTL.TXFFLSH = 1
   - USB_GRSTCTL.RXFFLSH = 1

The application must wait until the TXFFLSH bit and the RXFFLSH bits are cleared before performing any operation on the core.

38.4.5.1.1.2 Tx FIFO Operation With Thresholding

1. Receive FIFO RAM allocation

   RAM for Setup Packets: 7 * n + 6 locations must be reserved in the receive FIFO to receive up to n setup packets on control endpoints, where n is the number of control endpoints supported by the device core. Locations reserved for setup packets are not used by the core to write any other data.

   One location for Global OUT NAK

   It is recommended to have an Rx FIFO space for two thresholds. Along with each received threshold, a status information is also written to the FIFO. With the last threshold of a packet, two status DWORDs are written to the FIFO. With the last packet of a transfer, transfer complete status information is written to the FIFO. Thus, in the worst case, a minimum of 2 * (Rx_threshold length / 4 + 4) space must be allocated to receive a packet.
2. Transmit FIFO RAM allocation:

   The minimum RAM space required for each IN Endpoint Transmit FIFO is min (2 * TXTHRLEN, endpoint_max_pkt_size). More space allocated in the transmit IN Endpoint FIFO results in better performance on the USB and can hide latencies on the AHB.
3. Internal register storage space allocation.

   Same as in non-threshold mode of operation.

If high AHB latencies results in frequent underrun errors, the host may possibly disable this endpoint because of errors in the packet (typically when there is error in three consecutive packets). So thresholding must be enabled only when AHB latency is not very high.

38.4.5.1.2 Host Mode

Considerations for allocating data RAM for Host Mode FIFOs are listed here:

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (Largest Packet Size / 4) + 2 must be allotted to receive packets. If a high-bandwidth channel is enabled, or multiple isochronous channels are enabled, then at least two (Largest Packet Size / 4) + 2 spaces must be allotted to receive back-to-back packets. Typically, two (Largest Packet Size / 4) + 2 spaces are recommended so that when the previous packet is being transferred to AHB, the USB can receive the subsequent packet. If AHB latency is high, you must allocate enough space to receive multiple packets.

Along with each host channel's last packet, information on transfer complete status and channel halted is also pushed to the FIFO. So two locations must be allocated for this.

For handling NAK in DMA mode, the application must determine the number of Control/Bulk OUT endpoint data that must fit into the TX_FIFO at the same instant. Based on this, one location each is required for Control/Bulk OUT endpoints.

For example, when the host addresses one Control OUT endpoint and three Bulk OUT endpoints, and all these must fit into the non-periodic TX_FIFO at the same time, then four extra locations are required in the RX FIFO to store the rewind status information for each of these endpoints.

Transmit FIFO RAM allocation

The minimum amount of RAM required for the Host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB and can hide AHB latencies. Typically, two Largest Packet Sizes' worth of space is recommended, so that when the current packet is under transfer to the USB, the AHB can get the next packet. If the AHB latency is large, then you must allocate enough space to buffer multiple packets.

The minimum amount of RAM required for Host periodic Transmit FIFO is the largest maximum packet size among all supported periodic OUT channels. If there is at lease one High Bandwidth Isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

38.4.5.1.2.1 Internal Register Storage Space Allocation

When operating in DMA mode, the DMA address register for each host channel (USB_HCx_DMAADDR) is stored in the FIFO RAM. One location for each channel must be reserved for this.

FIFO Name	Data RAM Size
Receive Data FIFO	rx_fifo_size
Non-periodic Transmit FIFO	tx_fifo_size[0]
IN Endpoint Transmit FIFO	tx_fifo_size[1]

With this information, the following registers must be programmed:

1. Receive FIFO Size Register (USB_GRXFSIZ)
   - USB_GRXFSIZ.RXFDEP = rx_fifo_size;
2. Non-periodic Transmit FIFO Size Register (USB_GNPTXFSIZ)
   - USB_GNPTXFSIZ.NPTXFDEP = tx_fifo_size[0];
   - USB_GNPTXFSIZ.NPTXFSTADDR = rx_fifo_size;
3. Host Periodic Transmit FIFO Size Register (USB_HPTXFSIZ)
   - USB_HPTXFSIZ.PTXFSIZE = tx_fifo_size[1];
   - USB_HPTXFSIZ.PTXFSTADDR = USB_GNPTXFSIZ.NPTXFSTADDR + tx_fifo_size[0];
4. The transmit FIFOs and receive FIFO must be flushed after RAM allocation for proper FIFO function.
   - USB_GRSTCTL.TXFNUM = 0x10
   - USB_GRSTCTL.TXFFLSH = 1
   - USB_GRSTCTL.RXFFLSH = 1
   - The application must wait until the TXFFLSH bit and the RXFFLSH bits are cleared before performing any operation on the core.

38.4.5.1.3 Summary of Guidelines for Choosing Data FIFO RAM Depth in Host Mode

38.4.5.1.3.1 RX FIFO Size

The RX FIFO size must be equal to at least twice the largest value of MPS size used. The recommended minimum RXFIFO depth = ((largest packet size/4)*2)+2. (+2) is required by the core for the status quadlets internally.

38.4.5.1.3.2 Non Periodic TX FIFO Size

This should be equal to at least twice the largest value of MPS size used. The recommended minimum non-periodic TXFIFO depth = ((largest packet size/4)*2).

38.4.5.1.3.3 Periodic TX FIFO Size

The recommended size for Periodic TXFIFO is sum total of (MPS*MC)/4 for all the channels.

Note: In the above recommendations, always round off the MPS value to the nearest multiple of 4. For example, if the largest value of MPS=125, use the rounded-off value, which is 128.

38.4.5.1.4 Calculating the Total FIFO Size

1. 38.4.5.1.4.1 Thresholding Disabled
2. 38.4.5.1.4.2 Thresholding Enabled

38.4.5.1.4.1 Thresholding Disabled

The RxFIFO is shared between the host and device. The Host TxFIFOs are also shared with Device IN endpoint TxFIFOs 0 through n.

There are three ways to calculate the total FIFO size.

Method 1

Use this method if you are using the following conditions:

- Minimum FIFO depth allocation
- The FIFO must equal at least one MaxPacketSize (MPS).

Device RxFIFO =

- (4 * number of control endpoints + 6) + ((largest USB packet used / 4) + 1 for status information) + (2 * number of OUT endpoints) + 1 for Global NAK

Note: Include the Control OUT endpoint in the number of OUT endpoints.

Host RxFIFO =

- Slave mode

  Minimum requirement: (largest USB packet used / 4) + 1 for status information + 1 transfer complete
- DMA mode

  (largest USB packet used / 4) + 1 for status information + 1 transfer complete + 1 location each bulk/control out endpoint for handling NAK scenario

Host Non-Periodic TxFIFO =

- largest non-periodic USB packet used / 4

Host Periodic TxFIFO =

- Sum total of (MPS*MC)/4 of all periodic channels or 1500 locations, whichever is lower.

Device IN Endpoint TxFIFOs (a separate FIFO is allocated to each IN endpoint) =

- IN Endpoints Max packet Size / 4

Method 2

Use this method if you are using the recommended minimum FIFO depth allocation with support for high-bandwidth endpoints. This FIFO allocation enables the core to transfer a packet on the USB while the previous (next) packet is simultaneously transferred to the AHB. This FIFO allocation improves the core's performance.

Device RxFIFO =

- (4 * number of control endpoints + 6) + 2 * ((largest USB packet used / 4) + 1) +(2 * number of OUT endpoints) + 1

Host RxFIFO =

- Slave mode

  2 * ((largest USB packet used / 4) + 1 + 1)
- DMA mode

  2 * ((largest USB packet used / 4) + 1 + 1) + 1 location each bulk/control out endpoint for handling NAK scenario

Host Non-Periodic TxFIFO =

- 2 * (largest non-periodic USB packet used / 4)

Host Periodic TxFIFO =

- Sum total of (MPS*MC)/4 for all periodic channels or 1500 location, whichever is lower.

Device IN Endpoint-Specific TxFIFOs (a separate FIFO is allocated to each endpoint) =

- 2 * (max_pkt_size for the endpoint) / 4.

```
//DMA mode

      OTG Total RAM = (Device RxFIFO or Host RxFIFO; choose the largest one) +
```

```
      ((Host Non-Periodic TxFIFO + Host peiodic TxFIFO) or
       (Device IN Endpoint TxFIFO #0 + #1 + #2 + #n)); choose the largest one +
      (1 location per Host channel or 1 location per Device Endpoint direction; choose
       the largest one)

//Slave mode

    OTG Total RAM = (Device RxFIFO or Host RxFIFO; choose the largest one) +
    ((Host Non-Periodic TxFIFO + Host peiodic TxFIFO) or
     (Device IN Endpoint TxFIFO #0 + #1 + #2 + #n)); choose the largest one
```

Method 3

Use this method if you are using the recommended FIFO depth allocation that supports high-bandwidth endpoints and high AHB latency.

Note:

- x = (AHB latency + time to transfer largest packet on AHB) / time to transfer largest packet on USB.
- The value of x is an integer. Any fractional value is rounded to the nearest integer. For example: x = 20 ms / 17,039 ms = 1.17 ms = 2 ms.

Device RxFIFO =

- (4 * number of control endpoints + 6) + (x + 1) * ((largest USB packet used / 4) + 1)+ (2 * number of OUT endpoints) + 1

Note: Include the Control OUT endpoint in the number of OUT endpoints.

Host RxFIFO =

- Slave mode

  (x + 1) * ((largest USB packet used / 4) + 1 + 1)
- DMA mode

  (x + 1) * ((largest USB packet used / 4) + 1 + 1) + 1 location each bulk/control out endpoint for handling NAK scenario

Host Non-Periodic TxFIFO =

- (x + 1) * (largest non-periodic USB packet used / 4)

Host Periodic TxFIFO =

- (x+1) * (Sum total of (MPS*MC)/4 of all periodic channels or 1500 locations, whichever is lower).

Device IN Endpoint-Specific TxFIFOs (a separate FIFO is allocated to each endpoint) =

- (x+1)*(max_pkt_size for the endpoint)/4

```
//DMA mode
      OTG Total RAM = (Device RxFIFO or Host RxFIFO; choose the largest one) +
      ((Host Non-Periodic TxFIFO + Host periodic TxFIFO) OR
       (Device IN Endpoint TxFIFO #0 + #1 + #2 + #n); choose the largest one) +
      (1 location per Host channel or 1 location per Device Endpoint direction; choose
      the largest one)

//Slave mode
    OTG Total RAM = (Device RxFIFO or Host RxFIFO; choose the largest one) +
    ((Host Non-Periodic TxFIFO + Host periodic TxFIFO) OR
     (Device IN Endpoint TxFIFO #0 + #1 + #2 + #n); choose the largest one)
```

38.4.5.1.4.2 Thresholding Enabled

The main intention of threshold support are the following:

- To have a smaller FIFO size
- To have faster DMA response.

Thresholding is supported only in device mode. So if your device does not have many IN endpoints (not more than 2) OR if your calculated total device FIFO depth without thresholding is not more than the required host FIFO depth, then you are not going to save FIFO space much by enabling thresholding.

Device RxFIFO =

- (7 * number of control endpoints + 6) + 2 * ((min(largest USB packet used, 4 * USB_DTHRCTL.RXTHRLEN) / 4) + 4) + 1 for Global NAK.

Host RxFIFO =

- Slave mode

  (largest USB packet used / 4) + 1 for status information + 1 transfer complete
- DMA mode

  (largest USB packet used / 4) + 1 for status information + 1 transfer complete + 1 location each bulk/control out endpoint for handling NAK scenario

Host Non-Periodic TxFIFO =

- Largest non-periodic USB packet used / 4

Host Periodic TxFIFO =

- Sum total of (MC*MPS)/4 of all periodic channels or 1500 locations, whichever is lower.

Device IN Endpoint TxFIFOs (a separate FIFO is allocated to each IN endpoint) =

- min(2 x Transmit Threshold size, IN Endpoints Max packet Size) / 4

```
          OTG Total RAM = (Device RxFIFO or Host RxFIFO; choose the largest one) +
((Host Non-Periodic TxFIFO + Host peiodic TxFIFO) OR Device IN Endpoint
  TxFIFO #0 + #1 + #2 + #n); choose the largest one+ (1 location per
Host channel or 1 location per Device Endpoint direction; choose the largest
one)
```

38.4.5.2 Dynamic FIFO Allocation

The application can change the RAM allocation for each FIFO during the operation of the core.

The Core Interrupt Handler

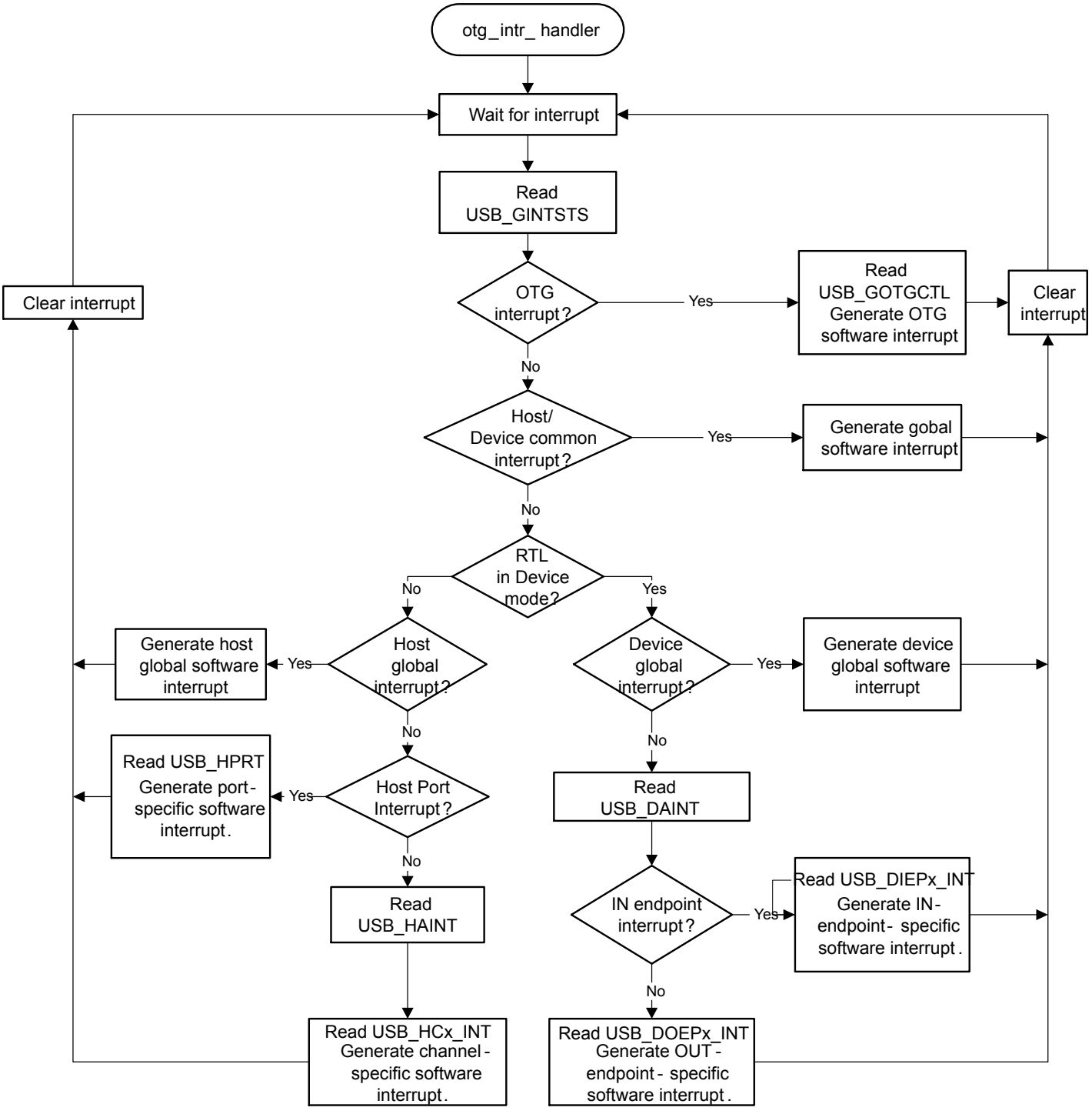


Figure 38.31. Core Interrupt Handler

38.4.5.2.1 Host Mode

In Host mode, before changing FIFO data RAM allocation, the application must determine the following.

All channels are disabled
- All FIFOs are empty

Once these conditions are met, the application can reallocate FIFO data RAM as explained in 38.4.5.1 Data FIFO RAM Allocation.

After reallocating the FIFO data RAM, the application must flush all FIFOs in the core using the USB_GRSTCTL.TXFFLSH (TxFIFO Flush) and USB_GRSTCTL.RXFFLSH (RxFIFO Flush) fields. Flushing is required to reset the pointers in the FIFOs for proper FIFO operation after reallocation. For more information on flushing FIFOs, see 38.4.5.2.3 Flushing TxFIFOs in the Core and 38.4.5.2.4 Flushing RxFIFOs in the Core.

38.4.5.2.2 Device Mode

In Device mode, before changing FIFO data RAM allocation, the application must determine the following.
- All IN and OUT endpoints are disabled
- NAK mode is enabled in the core on all IN endpoints
- Global OUT NAK mode is enabled in the core
- All FIFOs are empty

Once these conditions are met, the application can reallocate FIFO data RAM as explained in 38.4.5.1 Data FIFO RAM Allocation. When NAK mode is enabled in the core, the core responds with a NAK handshake on all tokens received on the USB, except for SETUP packets.

After the reallocating the FIFO data RAM, the application must flush all FIFOs in the core using the USB_GRSTCTL.TXFFLSH (TxFIFO Flush) and USB_GRSTCTL.RXFFLSH (RxFIFO Flush) fields. Flushing is required to reset the pointers in the FIFOs for proper FIFO operation after reallocation. For more information on flushing FIFOs, see 38.4.5.2.3 Flushing TxFIFOs in the Core and 38.4.5.2.4 Flushing RxFIFOs in the Core.

38.4.5.2.3 Flushing TxFIFOs in the Core

The application can flush all TxFIFOs in the core using USB_GRSTCTL.TXFFLSH as follows:
1. Check that USB_GINTSTS.GINNAKEFF=0. If this bit is cleared then set USB_DCTL.SGNPINNAK=1.
2. Wait for USB_GINTSTS.GINNAKEFF=1, which indicates the NAK setting has taken effect to all IN endpoints.
3. Poll USB_GRSTCTL.AHBIDLE until it is 1.

   AHBIdle = H indicates that the core is not writing anything to the FIFO.
4. Check that USB_GRSTCTL.TXFFLSH =0. If it is 0, then write the TxFIFO number you want to flush to USB_GRSTCTL.TXFNUM.
5. Set USB_GRSTCTL.TXFFLSH=1and wait for it to clear.
6. Set the USB_DCTL.GCNPINNAK bit.

38.4.5.2.4 Flushing RxFIFOs in the Core

The application can flush all RxFIFOs in the core using USB_GRSTCTL.RXFFLSH as follows:
1. Check the status of the USB_GINTSTS.GOUTNAKEFF bit. If it has been cleared, then set USB_DCTL.SGOUTNAK=1. Else, clear USB_GINTSTS.GOUTNAKEFF.

   NAK Effective interrupt = 1 indicates that the core is not writing to FIFO.
2. Wait for USB_GINTSTS.GOUTNAKEFF=1, which indicates the NAK setting has taken effect to all OUT endpoints.
3. Poll the USB_GRSTCTL.AHBIDLE until it is 1.

   AHBIDLE = 1 indicates that the core is not reading anything from the FIFO.
4. Set USB_GRSTCTL.RXFFLSH=1 and wait for it to clear.
5. Set the USB_DCTL.GCOUTNAK bit.

38.4.6 Suspend/Resume and SRP

This chapter describes different methods of saving power when the USB is suspended. This chapter discusses the following topics:

38.4.6.1 Placing PHY in Low Power Mode Without Entering Suspend

The core can place the PHY in low power mode (the differential receiver is disabled) without entering suspend.

38.4.6.1.1 When the Core is in Host Mode

Programming flow for the Host Core to put PHY in low power mode

1. To turn off port power, perform write operation to set the following bits in the USB_HPRT register:
   - USB_HPRT.PRTPWR = 0;
   - USB_HPRT.PRTENA = 0;
2. To put PHY in low power mode, perform read-modify-write operation to set the following bits in the USB_PCGCCTL register:
   - USB_PCGCCTL.STOPPCLK = 1
   - USB_PCGCCTL.GATEHCLK = 0

Programming flow for the Host Core to make PHY exit low power mode

If your device is non-SRP capable, the host must implement polling to detect the device connection by turning on the port and exiting PHY low power mode periodically and checking for connect.

1. To turn on port power, perform write operation to set the following bits in the USB_HPRT register:
   - USB_HPRT.PRTPWR = 1
   - USB_HPRT.PRTENA = 0
2. To exit PHY low power mode, perform read-modify-write operation to set the following bits in the USB_PCGCCTL register:
   - USB_PCGCCTL.STOPPCLK = 0
   - USB_PCGCCTL.STOPHCLK = 0
3. Wait for the USB_HPRT Port Connect Detected (PRTCONNDET) bit to be set and do the enumeration of the device.

If your device is SRP-capable, when the device initiates SRP request, the Host core asynchronously detects SRP and the PHY exits low power mode.

1. Wait for Session Request from the device, or New Session Detected Interrupt (SESSREQINT) in the USB_GINTSTS register.
2. To turn on port power, perform write operation to set the following bits in the USB_HPRT register:
   - USB_HPRT.PRTPWR = 1
   - USB_HPRT.PRTENA = 0
3. Wait for the USB_HPRT Port Connect Detected (PRTCONNDET) bit to be set and do the enumeration of Device.

38.4.6.1.2 When the Core is in Device Mode

To make PHY enter low power mode, complete the following steps:

1. Ensure that the following signals are set as follows:
   - VBUS voltage level must be below the session valid level (VBUS is not active)
   - DP/DM must be SE0
2. From the application, perform read-modify-write operation to set USB_PCGCCTL.STOPPCLK = 1.

38.4.6.2 Suspend

When the core is in Suspend, the following power conservation options are available to use:

- 38.4.6.2.1 Using EM2: You can enter EM2, turning off power (and reseting) parts of the core
- 38.4.6.2.2 Using Clock Gating in EM0/EM1: You can choose gate the AHB clock to some parts of the core 38.4.6.2.2.1 Internal Clock Gating When the Core is in Host Mode

This section discusses methods of conserving power by using one of the above methods.

38.4.6.2.1 Using EM2

38.4.6.2.1.1 Overview of the EM2 Programming Model

When the USB is suspended or the session is not valid, the PHY is driven into Suspend mode, stopping the PHY clock to reduce power consumption in the PHY and the core. To further reduce power consumption, the core also supports AHB clock gating and using EM2.

The following sections show the procedures you must follow to use EM2 while in suspend/session-off.

During EM2, the clock to the core must be switched to one of the 32 kHz sources (LFRCO or LFXO). This core needs this clock to detect Resume and SRP events.

38.4.6.2.1.2 EM2 When the Core is in Host Mode

Host Mode Suspend in EM2

Sequence of operations:

1. Back up the essential registers of the core. Read and store the following core registers:
   - USB_GINTMSK
   - USB_GOTGCTL
   - USB_GAHBCFG
   - USB_GUSBCFG
   - USB_GRXFSIZ
   - USB_GNPTXFSIZ
   - USB_DCFG
   - USB_DCTL
   - USB_DAINTMSK
   - USB_DIEPMSK
   - USB_DOEPMSK
   - USB_DIEPx_CTL
   - USB_DIEPx_TSIZ
   - USB_DIEPx_DMAADDR
   - USB_PCGCCTL
   - USB_DIEPTXFn
2. The application sets the Port Suspend bit in the Host Port CSR, and the core drives a USB suspend.
3. The application sets the Power Clamp bit in the Power and Clock Gating Control register.
4. The application sets the Reset to Power-Down Modules bit in the Power and Clock Gating Control register.
5. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core suspends the PHY and the PHY clock stops. If USB_HCFG.ENA32KHZS is set, switch the USBC clock to 32 kHz.
6. Enter EM2.

Host Mode Resume in EM2

Sequence of operations:

1. The resume event starts by the application waking up from EM2 (on an interrupt)
2. Switch USBC clock back to 48 MHz.
3. The application clears the Stop PHY Clock bit and the core takes the PHY back to normal mode. The PHY clock starts up.
4. The application clears the Power Clamp bit. The core starts driving Resume signaling on the USB.
5. The application clears the Reset to Power-Down Modules bit.
6. The application programs registers in the CSR and sets the Port Resume bit in Host Port CSR (Setting the Port Resume bit is required by the core, although Resume signaling starts earlier).
7. The application clears the Port Resume bit and the core stops driving Resume signaling.

The core is in normal operating mode.

Note: The application must insert delays of at least 2 PHY clocks between all steps in this sequence. This requirement applies to all USB EM2 programming sequences.

Host Mode Remote Wakeup in EM2

Sequence of operations:

1. The core detects Remote Wakeup signaling on the USB. The PHY exits suspend mode and the PHY clock restarts.
2. The core generates a Remote Wakeup Detected interrupt. The Remote Wakeup interrupt is generated using the 32 kHz clock depending on the USB_HCFG.RESVALID (ResumeValidPeriod) programmed. The Host Core starts resume signaling at this stage.
3. The USBC clock is switched back to normal 48 MHz clock.
4. The application clears the Stop PHY Clock bit.
5. The application clears the Power Clamp bit.
6. The application clears the Reset to Power-Down Modules bit
7. The application programs CSRs and sets the Port Resume bit. The core continues to drive Resume signaling on the USB.
8. The application clears the Port Resume bit and the core stops driving Resume signaling.

The core enters normal operating mode.

Host Mode Session End in EM2

Sequence of operations:

1. Back up the essential registers of the core. Read and store the following core registers:
   - USB_GINTMSK
   - USB_GOTGCTL
   - USB_GAHBCFG
   - USB_GUSBCFG
   - USB_GRXFSIZ
   - USB_GNPTXFSIZ
   - USB_DCFG
   - USB_DCTL
   - USB_DAINTMSK
   - USB_DIEPMSK
   - USB_DOEPMSK
   - USB_DIEPx_CTL
   - USB_DIEPx_TSIZ
   - USB_DIEPx_DMAADDR
   - USB_PCGCCTL
   - USB_DIEPTXFn
2. The application sets the Port Suspend bit in the Host Port CSR and the core drives a USB suspend.
3. The application clears the Port Power bit.
4. The application sets the Power Clamp bit in the Power and Clock Gating Control register, and the core clamps the signals between the internal modules on different power rails.
5. The application sets the Reset to Power-Down Modules bit in the Power and Clock Gating Control register.
6. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, and the core suspends the PHY, stopping the PHY clock.
7. Switch USBC clock to 32 kHz.
8. Enter EM2.

Host Mode Session Start (EM2 -> EM0)

Sequence of operations:

1. Exit EM2/Enter EM0).
2. Switch USBC clock back to 48 MHz.
3. The application clears the Stop PHY Clock bit.
4. The application clears the Power Clamp bit. The application clears the Reset to Power-Down Modules bit.
5. The application programs CSRs and sets the Port Power bit to turn on VBUS.
6. The core detects the connection and drives the USB reset.

The core enters normal operating mode.

Host Mode Session End (EM0 -> EM2)

Sequence of operations:

1. Back up the essential registers of the core. Read and store the following core registers:
   - USB_GINTMSK
   - USB_GOTGCTL
   - USB_GAHBCFG
   - USB_GUSBCFG
   - USB_GRXFSIZ
   - USB_GNPTXFSIZ
   - USB_DCFG
   - USB_DCTL
   - USB_DAINTMSK
   - USB_DIEPMSK
   - USB_DOEPMSK
   - USB_DIEPx_CTL
   - USB_DIEPx_TSIZ
   - USB_DIEPx_DMAADDR
   - USB_PCGCCTL
   - USB_DIEPTXFn
2. The application sets the Port Suspend bit in the Host Port CSR and the core drives a USB suspend.
3. The application clears the Port Power bit.
4. The application sets the Power Clamp bit in the Power and Clock Gating Control register, and the core clamps the signals between the internal modules on different power rails.
5. The application sets the Reset to Power-Down Modules bit in the Power and Clock Gating Control register.
6. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register.
7. Enter EM2.

Host Mode Sessions Start (SRP) (EM2 -> EM0)

Sequence of operations:

1. The core detects SRP (data line pulsing) on the bus. The core de-asserts the suspend_n signal to the PHY, generating the PHY clock. The SRP Detected interrupt is generated.
2. The application clears the Stop PHY Clock bit, the core deasserts the suspend_n signal to the PHY to generate the PHY clock.
3. The power (VDD_DN) is turned on and stabilizes.
4. The application clears the Power Clamp bit.
5. The application clears the Reset to Power-Down Modules bit.
6. The application programs the CSRs, and sets the Port Power bit to turn on VBUS.
7. The core detects device connection and drives a USB reset.

The core enters normal operating mode.

38.4.6.2.1.3 EM2 When the Core is in Device Mode

Device Mode Suspend With EM2

In Device mode, the device validates the host-driven Resume signal for a period of 1.5 µs (75 clock cycles at 48 MHz). With a 32-KHz clock, 2.34 ms is required (75 clock cycles at 32 KHz) to detect the resume. Hence, the application programs USB_DCFG.RESVALID with a value of 4 clock cycles (125 µs). If the core is in Suspend mode, the device thus detects the resume and the host signals a resume for a minimum of 125 µs.

If the device is being reset from suspend, it begins a high-speed detection handshake after detecting SE0 for no fewer than 2.5 µs. With a 48-MHz clock, detection occurs after 120 clock cycles (2.5 µs). With a 32-kHz clock, 120 clock cycles signifies 3.75 msec. Hence, a programmable value of 4 clock cycles (125 µs) is used to detect reset.

The 32-KHz Suspend feature incorporates switching to the 32-KHz clock during suspend and resume/remote wakeup until the system comes up and starts driving 48 MHz.

Sequence of operations:

1. Detect Suspend state. Wait for an interrupt from the device core and check that USB_GINTSTS.USBSUSP is set to 1.
2. Back up the essential registers of the core. Read and store the following core registers:
   - USB_GINTMSK
   - USB_GOTGCTL
   - USB_GAHBCFG
   - USB_GUSBCFG
   - USB_GRXFSIZ
   - USB_GNPTXFSIZ
   - USB_DCFG
   - USB_DCTL
   - USB_DAINTMSK
   - USB_DIEPMSK
   - USB_DOEPMSK
   - USB_DIEPx_CTL
   - USB_DIEPx_TSIZ
   - USB_DIEPx_DMAADDR
   - USB_PCGCCTL
   - USB_DIEPTXFn
3. The application sets the PWRCLMP bit in the Power and Clock Gating Control (USB_PCGCCTL) register.
4. The application sets the USB_PCGCCTL.RSTPDWNMODULE bit.
5. The application sets the USB_PCGCCTL.STOPPCLK bit.
6. Switch USBCLK to 32 kHz.
7. Enter EM2.

Device Mode Resume (EM2 -> EM0)

Sequence if operations:

1. The core detects Resume signaling on the USB. The core generates a Resume Detected interrupt.
2. Switch USBCLK back to 48 MHz.
3. The application clears the STOPPCLK bit.
4. The application clears the USB_PCGCCTL.PWRCLMP and USB_PCGCCTL.RSTPDWNMODULE bits.
5. Restore the USB_GUSBCFG and USB_DCFG registers with the values stored during the Save operation before entering EM2.
6. Restore the following core registers with the values stored during the Save operation before entering EM2:
   - USB_GINTMSK
   - USB_GOTGCTL
   - USB_GUSBCFG
   - USB_GRXFSIZ
   - USB_GNPTXFSIZ
   - USB_DAINTMSK
   - USB_DIEPMSK
   - USB_DOEPMSK
   - USB_DIEPx_CTL
   - USB_DIEPx_TSIZ
   - USB_DIEPx_DMAADDR
   - USB_DIEPTXFn
7. The application programs CSRs, then sets the Power-On Programming Done bit in the Device Control register.

Device Mode Remote Wakeup (EM2 -> EM0)

Sequence if operations:

1. An interrupt wakes up the device from EM2.
2. Switch USBCLK (USBC) back to 48 MHz.
3. The application clears the STOPPCLK and GATEHCLK bits in the USB_PCGCCTL register.
4. The application clears the USB_PCGCCTL.PWRCLMP and USB_PCGCCTL.RSTPDWNMODULE bits.

5. Restore the USB_GUSBCFG and USB_DCFG registers with the values stored during the Save operation before entering EM2 .
6. Drive remote wakeup from the core. Program USB_DCTL by performing write-only operation with the following values:
   - USB_DCTL.RMTWKUPSIG = 1
   - Other Bits = Value stored during the Save operation before entering EM2
7. Clear all interrupt status. Wait for at least 1 millisecond of remote wakeup time and then program GINSTS register with 0xFFFFFFFF to clear all the status register fields.
8. Restore the following core registers with the values stored during the Save operation before entering EM2:
   - USB_GINTMSK
   - USB_GOTGCTL
   - USB_GUSBCFG
   - USB_GRXFSIZ
   - USB_GNPTXFSIZ
   - USB_DAINTMSK
   - USB_DIEPMSK
   - USB_DOEPMSK
   - USB_DIEPx_CTL
   - USB_DIEPx_TSIZ
   - USB_DIEPx_DMAADDR
   - USB_DIEPTXFn
9. Wait for remote wakeup time (1-15ms) and then program USB_DCTL by performing read-modify-write to set USB_DCTL.RMTWKUPSIG = 0.

Device Mode Session End (EM0 -> EM2)

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt. The host turns off VBUS.
2. The application sets the Power Clamp bit in the Power and Clock Gating Control register.
3. The application sets the Reset to Power-Down Modules bit in the Power and Clock Gating Control register.
4. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register.
5. Switch USBCLK to 32 kHz.
6. Enter EM2.

Device Mode Session Start (EM2 -> EM0)

Sequence of operations:

1. The core detects VBUS on (voltage level within session-valid). A New Session Detected interrupt is generated.
2. Switch USBCLK (USBC) back to 48 MHz.
3. The application clears the Stop PHY Clock bit.
4. The application clears the Power Clamp bit.
5. The application clears the Reset to Power-Down Modules bit.
6. The application programs CSRs.
7. The cores detects a USB reset.

The core enters normal operating mode.

38.4.6.2.2 Using Clock Gating in EM0/EM1

The core supports HCLK gating to reduce dynamic power to internal modules to the core during Suspend/session-off state in EM0 and EM1.

38.4.6.2.2.1 Internal Clock Gating When the Core is in Host Mode

The following sections show the procedures you must follow to use the clock gating feature.

Host Mode Suspend and Resume With Clock Gating

Sequence of operations:

1. The application sets the Port Suspend bit in the Host Port CSR, and the core drives a USB suspend.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register. The application sets the Gate hclk bit in the Power and Clock Gating Control register, the core gates the hclk internally.
3. The core remains in Suspend mode.
4. The application clears the Gate hclk and Stop PHY Clock bits, and the PHY clock is generated.
5. The application sets the Port Resume bit, and the core starts driving Resume signaling.
6. The application clears the Port Resume bit after at least 20 ms.
7. The core is in normal operating mode.

Host Mode Suspend and Remote Wakeup With Clock Gating

Sequence of operations:

1. The application sets the Port Suspend bit in the Host Port CSR, and the core drives a USB suspend.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates hclk internally.
3. The core remains in Suspend mode
4. The Remote Wakeup signaling from the device is detected. The core generates a Remote Wakeup Detected interrupt.
5. The application clears the Gate hclk and Stop PHY Clock bits. The core sets the Port Resume bit.
6. The application clears the Port Resume bit after at least 20 ms.
7. The core is in normal operating mode.

Host Mode Session End and Start With Clock Gating

Sequence of operations:

1. The application sets the Port Suspend bit in the Host Port CSR, and the core drives a USB suspend.
2. The application clears the Port Power bit. The core turns off VBUS.
3. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates hclk internally.
4. The core remains in Low-Power mode.
5. The application clears the Gate hclk bit and the application clears the Stop PHY Clock bit to start the PHY clock.
6. The application sets the Port Power bit to turn on VBUS.
7. The core detects device connection and drives a USB reset.
8. The core is in normal operating mode.

Host Mode Session End and SRP With Clock Gating

Sequence of operations:

1. The application sets the Port Suspend bit in the Host Port CSR, and the core drives a USB suspend.
2. The application clears the Port Power bit. The core turns off VBUS.
3. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates hclk internally.
4. The core remains in Low-Power mode.
5. SRP (data line pulsing) from the device is detected. An SRP Request Detected interrupt is generated.
6. The application clears the Gate hclk bit and the Stop PHY Clock bit.
7. The core sets the Port Power bit to turn on VBUS.
8. The core detects device connection and drives a USB reset.
9. The core is in normal operating mode.

38.4.6.2.2.2 Internal Clock Gating When the Core is in Device Mode

The following sections show the procedures you must follow to use the clock gating feature.

Device Mode Suspend and Resume With Clock Gating

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates hclk.
3. The core remains in Suspend mode.
4. The Resume signaling from the host is detected. A Resume Detected interrupt is generated.
5. The application clears the Gate hclk bit and the Stop PHY Clock bit.
6. The host finishes Resume signaling.
7. The core is in normal operating mode.

Device Mode Suspend and Remote Wakeup With Clock Gating

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register. The application sets the Gate hclk bit in the Power and Clock Gating Control register, the core gates hclk.
3. The core remains in Suspend mode.
4. The application clears the Gate hclk bit and the Stop PHY Clock bit.
5. The application sets the Remote Wakeup bit in the Device Control register, the core starts driving Remote Wakeup signaling.
6. The host drives Resume signaling.
7. The core is in normal operating mode.

Device Mode Session End and Start With Clock Gating

Sequence of operations:

1. The core detects a USB suspend, and generates a Suspend Detected interrupt. The host turns off VBUS.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates hclk.
3. The core remains in Low-Power mode.
4. The new session is detected (A session-valid voltage is detected). A New Session Detected interrupt is generated.
5. The application clears the Gate hclk and Stop PHY Clock bits.
6. The core detects USB reset.
7. The core is in normal operating mode

Device Mode Session End and SRP With Clock Gating

Sequence of operations:

1. The core detects a USB suspend, and generates a Suspend Detected interrupt. The host turns off VBUS.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates hclk.
3. The core remains in Low-Power mode.
4. The application clears the Gate hclk and Stop PHY Clock bits.
5. The application sets the SRP Request bit, and the core drives data line and VBUS pulsing.
6. The host turns on VBUS, detects device connection, and drives a USB reset.
7. The core is in normal operating mode.

38.4.7 Register Usage

Only the Core Global, Power and Clock Gating, Data FIFO Access, and Host Port registers can be accessed in both Host and Device modes. When the core is operating in one mode, either Device or Host, the application must not access registers from the other mode. If an illegal access occurs, a Mode Mismatch interrupt is generated and reflected in the Core Interrupt register (USB_GINTSTS.MODEMIS).

When the core switches from one mode to another, the registers in the new mode must be reprogrammed as they would be after a power-on reset.

The memory map for the core is as follows:

- Core Global Registers are located in the address offset-range [0xDE000, 0xDE3FF] and typically start with first letter G.
- Host Mode Registers are located in the address offset-range [0xDE400, 0xDE7FF] and start with first letter H.
- Device Mode Registers are located in the address offset-range [0xDE800, 0xDEDFF] and start with first letter D.
- The Power and Clock Gating register is located at offset 0xDEE00.
- The Device EP/Host Channel FIFOs start at address offset 0xDF000 with 4K spacing. These registers, available in both Host and Device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.
- The Direct RAM Access area start at address offset 0xFE000.

38.5 Charger Detect Function

The USB module contains a charger detection block which is compliant with the USB-IF Battery Charging Specification, Revision 1.2. Upon establishing a physical connection to a USB host, the peripheral can distinguish between a standard downstream port (SDP), dedicated charging port (DCP), or a charging downstream port (CDP). ADC multiplexer connections to the USB D+ and D- pins are also provided internally for detecting the presence of non-standard charging hardware. Firmware may optionally implement algorithms to detect ACA or non-compliant charger hardware.

Note: The USB charger detect function only distinguishes between the various types of USB ports outlined in the specification. The device itself does not contain direct battery management or battery charging circuitry.

Firmware interfaces to the USB charger detection hardware through three special function registers: USB_CTRL, USB_CMD, and USB_STATUS. The USB_CTRL and USB_CMD registers configure and control the hardware, while USB_STATUS provides status information. The charger detection hardware shares an interrupt with the VBUS detection interrupt, allowing firmware to use the same interrupt service routine to handle all of the USB charger detect functions. Interrupts may be generated on the following events:

- VBUS detection
- VBUS removal (generates an error interrupt)
- Completion of data contact detection (DCD) phase
- Completion of primary detection (PD) phase
- Completion of secondary detection (SD) phase

Additionally, the charger detection block allows firmware to selectively choose which functions will be performed when charge detection is enabled. Data contact detection (DCD), DCD timeout, primary detection (PD), and secondary detection (SD) may all be enabled individually. Hardware does not perform any of these operations until the charger detection function is enabled using the USB_CMD.STARTCD bit and the hardware detects a valid VBUS signal. If VBUS is not enabled, it is assumed to be present by the hardware. Once USB_CMD.STARTCD is enabled, the hardware proceeds through the selected functions in the following order, skipping any that are not enabled:

- Data Contact Detection
- Primary Detection
- Secondary Detection

38.5.1 Charger Detection Flow

The figure Figure 38.32 Charger Detection Flow on page 1559 shows the Charger Detection Flow. The Charger Detection Flow requires firmware and hardware action to perform different functions of Charger Detection.

The Charger Detection flow allows for the detection of CDP, DCP and SDP ports which will be described in 38.5.2 Detection of SDP, DCP, and CDP

There are optional firmware action which allows for detection of non compliant Charger and ACA chargers. Non compliant charger detection is charger dependent, ACA charger detection is described in 38.5.3 Atypical Charger Detection

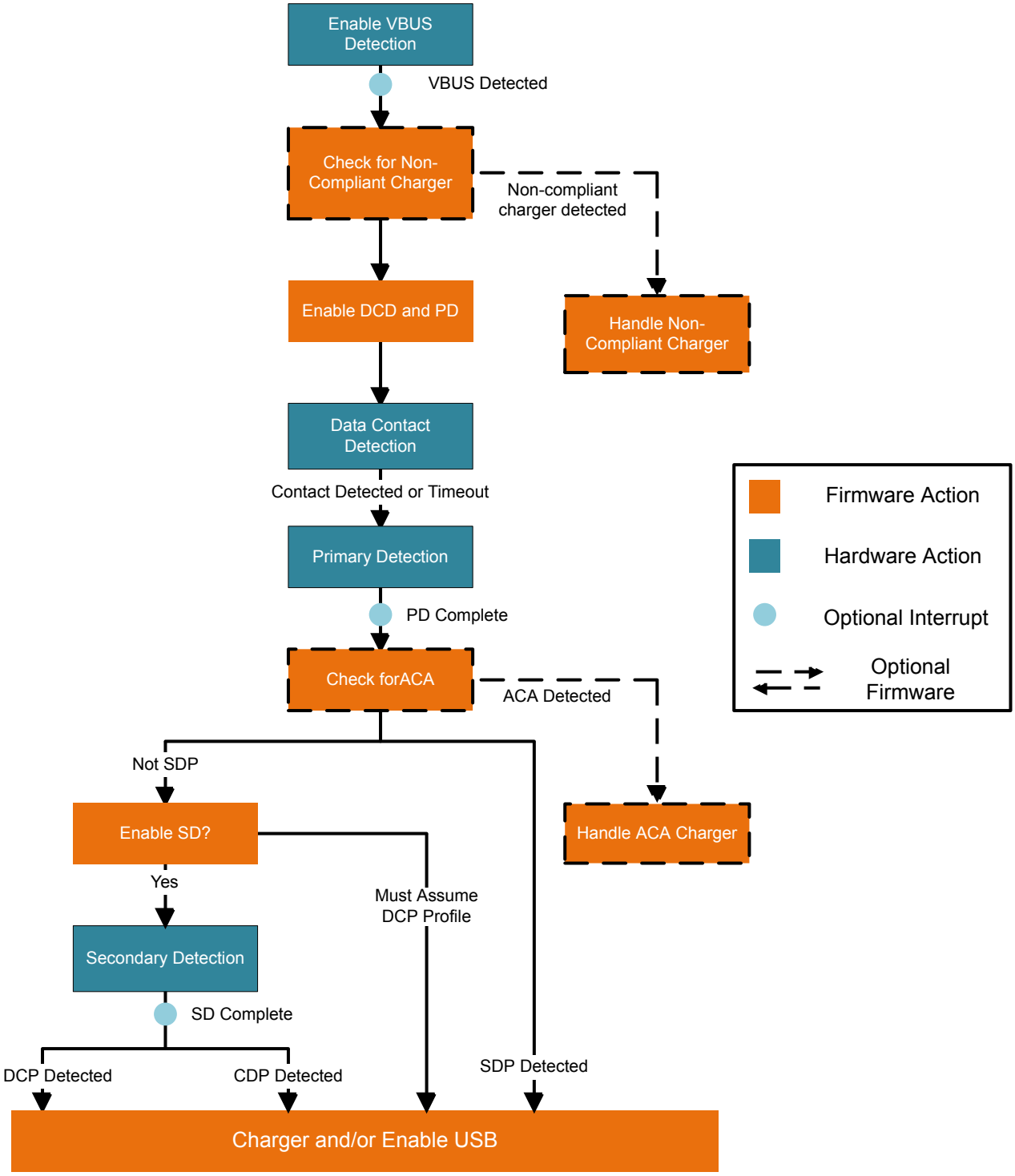


Figure 38.32. Charger Detection Flow

38.5.2 Detection of SDP, DCP, and CDP

The most common and straightforward usage of the charger detection block is to determine the type of USB port to which the device has been connected. Each type of port has different load profile, maximum current, and communications capabilities, per the specification. To use the charger detection block for this purpose:

1. Enable VBUS detection
   - Enable VBUS detection on the VBUS pin (must be connected to USB VBUS).
   - Enable the VBUS detection interrupt with the corresponding interrupt enable bit.
   - Wait for VBUS High Detection to occur (USB_IF.VBUSDETH = 1, or service the interrupt).
2. Check for Non Compliant Charger
   - The detection of Non Compliant Charger is charger dependent, and many can be detected by measuring the voltages of D+ and D- using the ADC
3. Enable DCD and PD
   - Optionally, enable the PD and/or SD interrupts with the corresponding interrupt enable bits.
   - Set USB_CTRL.DCDEN to "Full Detection" (0x3) to enable data contact detection and the associated timeout circuit.
   - Set USB_CTRL.PDEN to enable Primary Detection.
   - Set USB_CMD.STARTCD to begin the charge detect sequence.
4. Data Contact Detection
   - Hardware is running the Data Contact Detection
5. Primary Detection
   - Hardware is running the Primary Detection
   - Wait for Primary Detection to complete (USB_IF.PD = 1, or service the interrupt).
   - The USB_STATUS.SDP bit will indicate if a Standard Downstream Port is detected.
6. Check for ACA
   - This step is optional and should be included if ACA detection is required. Refer to 38.5.3 Atypical Charger Detection
7. Enable SD
   - If the application requires further differentiation between DCP and CDP, set USB_CTRL.SDEN to enable Secondary Detection, and set USB_CMD.STARTCD to begin this sequence.
8. Secondary Detection
   - Hardware is running the Secondary Detection
   - Wait for Secondary Detection to complete (USB_IF.SD = 1, or service the interrupt).
   - The DCP and CDP bits will indicate if a Dedicated Charging Port or Charging Downstream Port has been detected.

38.5.3 Atypical Charger Detection

It is possible to detect ACA chargers, as well as certain chargers that do not comply with the USB specification, using additional resources on the device.

38.5.3.1 ACA Charger Detection

Accessory charging adapters (ACA) chargers use a resistor to ground on a special ID pin and a specific voltage on the USB D- pin to encode the type of ACA and its capabilities. If ACA detection is required, the ID pin signal should be connected to ID input on the device, and the USB PHY ID current source must be enabled.

The ADC may be used to measure the voltage on the ID signal, and the status bits USB_STATUS.ACALS and USB_STATUS.LS can be checked after Primary Detection to distinguish between different ACA options. Applications needing to determine ACA ports should check for ACA after primary detection is complete.

The Battery Charging Specifications specifies the set of resistor values on the ID pin that determines the different ACA chargers. The resistor values can be determined from the voltage measured by the ADC.

- RID_FLOAT: If ID resistance = RID_FLOAT, Charger is CDP, DCP, SDP, else Charger is ACA
- RID_A: If ID resistance = RID_A, Charger is either ACA-Dock or ACA-A
- RID_B: If ID resistance = RID_B, Charger is ACA-B
- RID_C: If ID resistance = RID_C, Charger is ACA-C

To detect ACA chargers, after the Primary Detection action in the Charger Detection Flow, two additional Firmware action are required

1. Check for ACA
   - Enable USB PHY ID current source by setting USB_CTRL.IDCDEN = 1.
   - Set up ADC0 to measure voltage at ID pin.
   - If ID resistance = RID_FLOAT, ACA is not detected, proceed to detect for CDP, DCP and SDP
   - If ID resistance != RID_FLOAT, ACA is detected, proceed to handle ACA Charger
2. Handle ACA Charger
   - If ID resistance = RID_A and USB_STATUS.ACALS=1, USB_STATUS.ACAFS=0, ACA-A with Low-Speed B device is detected
   - If ID resistance = RID_A and USB_STATUS.ACALS=0, USB_STATUS.ACAFS=1, ACA-A with Full-Speed B device is detected
   - If ID resistance = RID_A and USB_STATUS.ACALS=0, USB_STATUS.ACAFS=0, ACA-Dock is detected
   - If ID resistance = RID_B, ACA-B is detected
   - If ID resistance = RID_C, ACA-C is detected

38.5.3.2 Non Compliant Detection

Many dedicated charging units pre-date the USB Battery Charging Specification or do not comply with this specification for other reasons, such as additional supply current capabilities . Most of these cases implement resistive voltage dividers to produce very specific voltages on the D+ and D- pins. In this case, the D+ and D- pins may be measured directly using the ADC to determine the voltage levels and whether such a charger is attached. Normally, this would be performed after VBUS is detected and before going through the data contact detection sequence.

38.6 Register Map

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

Offset	Name	Type	Description
0x000	USB_CTRL	RW	System Control Register
0x004	USB_STATUS	R	System Status Register
0x008	USB_IF	R	Interrupt Flag Register
0x00C	USB_IFS	W1	Interrupt Flag Set Register
0x010	USB_IFC	(R)W1	Interrupt Flag Clear Register
0x014	USB_IEN	RW	Interrupt Enable Register
0x018	USB_ROUTE	RW	I/O Routing Register
0x02C	USB_CDCONF	RW	Charger Detect Configuration Register
0x030	USB_CMD	W1	Command Register
0x034	USB_DATTRIM1	RW	Data TRIM 1 Values for USB DP and DM
0x03C	USB_LEMCTRL	RW	USB LEM Control Register
0x040	USB_ROUTELOC0	RW	I/O Routing Location Register
0xDE000	USB_GOTGCTL	RWH	OTG Control and Status Register
0xDE004	USB_GOTGINT	RW	OTG Interrupt Register
0xDE008	USB_GAHBCFG	RW	AHB Configuration Register
0xDE00C	USB_GUSBCFG	RWH	USB Configuration Register
0xDE010	USB_GRSTCTL	RWH	Reset Register
0xDE014	USB_GINTSTS	RWH	Interrupt Register
0xDE018	USB_GINTMSK	RW	Interrupt Mask Register
0xDE01C	USB_GRXSTSR	R	Receive Status Debug Read Register
0xDE020	USB_GRXSTSP	R	Receive Status Read /Pop Register
0xDE024	USB_GRXFSIZ	RW	Receive FIFO Size Register
0xDE028	USB_GNPTXFSIZ	RW	Non-periodic Transmit FIFO Size Register
0xDE02C	USB_GNPTXSTS	R	Non-periodic Transmit FIFO/Queue Status Register
0xDE040	USB_GSNPSID	R	Synopsys ID Register
0xDE05C	USB_GDFIFOCFG	RW	Global DFIFO Configuration Register

Offset	Name	Type	Description
0xDE100	USB_HPTXFSIZ	RW	Host Periodic Transmit FIFO Size Register
0xDE104	USB_DIEPTXF1	RW	Device IN Endpoint Transmit FIFO Size Register 1
0xDE108	USB_DIEPTXF2	RW	Device IN Endpoint Transmit FIFO Size Register 2
0xDE10C	USB_DIEPTXF3	RW	Device IN Endpoint Transmit FIFO Size Register 3
0xDE110	USB_DIEPTXF4	RW	Device IN Endpoint Transmit FIFO Size Register 4
0xDE114	USB_DIEPTXF5	RW	Device IN Endpoint Transmit FIFO Size Register 5
0xDE118	USB_DIEPTXF6	RW	Device IN Endpoint Transmit FIFO Size Register 6
0xDE400	USB_HCFG	RW	Host Configuration Register
0xDE404	USB_HFIR	RW	Host Frame Interval Register
0xDE408	USB_HFNUM	R	Host Frame Number/Frame Time Remaining Register
0xDE410	USB_HPTXSTS	R	Host Periodic Transmit FIFO/Queue Status Register
0xDE414	USB_HAINT	R	Host All Channels Interrupt Register
0xDE418	USB_HAINTMSK	RW	Host All Channels Interrupt Mask Register
0xDE440	USB_HPRT	RWH	Host Port Control and Status Register
0xDE500	USB_HC0_CHAR	RW	Host Channel x Characteristics Register
0xDE504	USB_HC0_SPLT	RW	Host Channel x Split Control Register
0xDE508	USB_HC0_INT	RW1	Host Channel x Interrupt Register
0xDE50C	USB_HC0_INTMSK	RW	Host Channel x Interrupt Mask Register
0xDE510	USB_HC0_TSIZ	RW	Host Channel x Transfer Size Register
0xDE514	USB_HC0_DMAADDR	RW	Host Channel x DMA Address Register
...	USB_HCx_CHAR	RW	Host Channel x Characteristics Register
...	USB_HCx_SPLT	RW	Host Channel x Split Control Register
...	USB_HCx_INT	RW1	Host Channel x Interrupt Register
...	USB_HCx_INTMSK	RW	Host Channel x Interrupt Mask Register
...	USB_HCx_TSIZ	RW	Host Channel x Transfer Size Register

Offset	Name	Type	Description
...	USB_HCx_DMAADDR	RW	Host Channel x DMA Address Register
0xDE6A0	USB_HC13_CHAR	RW	Host Channel x Characteristics Register
0xDE6A4	USB_HC13_SPLT	RW	Host Channel x Split Control Register
0xDE6A8	USB_HC13_INT	RW1	Host Channel x Interrupt Register
0xDE6AC	USB_HC13_INTMSK	RW	Host Channel x Interrupt Mask Register
0xDE6B0	USB_HC13_TSIZ	RW	Host Channel x Transfer Size Register
0xDE6B4	USB_HC13_DMAADDR	RW	Host Channel x DMA Address Register
0xDE800	USB_DCFG	RW	Device Configuration Register
0xDE804	USB_DCTL	RWH	Device Control Register
0xDE808	USB_DSTS	R	Device Status Register
0xDE810	USB_DIEPMSK	RW	Device IN Endpoint Common Interrupt Mask Register
0xDE814	USB_DOEPMSK	RW	Device OUT Endpoint Common Interrupt Mask Register
0xDE818	USB_DAINT	R	Device All Endpoints Interrupt Register
0xDE81C	USB_DAINTMSK	RW	Device All Endpoints Interrupt Mask Register
0xDE828	USB_DVBUSDIS	RW	Device VBUS Discharge Time Register
0xDE82C	USB_DVBUSPULSE	RW	Device VBUS Pulsing Time Register
0xDE830	USB_DTHRCTL	RW	Device Threshold Control Register
0xDE834	USB_DIEPEMPMSK	RW	Device IN Endpoint FIFO Empty Interrupt Mask Register
0xDE900	USB_DIEP0CTL	RWH	Device Control IN Endpoint 0 Control Register
0xDE908	USB_DIEP0INT	RWH	Device IN Endpoint 0 Interrupt Register
0xDE910	USB_DIEP0TSIZ	RW	Device IN Endpoint 0 Transfer Size Register
0xDE914	USB_DIEP0DMAADDR	RW	Device IN Endpoint 0 DMA Address Register
0xDE918	USB_DIEP0TXFSTS	R	Device IN Endpoint Transmit FIFO Status Register 0
0xDE920	USB_DIEP0_CTL	RWH	Device Control IN Endpoint x+1 Control Register

Offset	Name	Type	Description
0xDE928	USB_DIEP0_INT	RWH	Device IN Endpoint x+1 Interrupt Register
0xDE930	USB_DIEP0_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register
0xDE934	USB_DIEP0_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
0xDE938	USB_DIEP0_DTXFSTS	R	Device IN Endpoint Transmit FIFO Status Register 1
...	USB_DIEPx_CTL	RWH	Device Control IN Endpoint x+1 Control Register
...	USB_DIEPx_INT	RWH	Device IN Endpoint x+1 Interrupt Register
...	USB_DIEPx_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register
...	USB_DIEPx_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
...	USB_DIEPx_DTXFSTS	R	Device IN Endpoint Transmit FIFO Status Register 1
0xDE9C0	USB_DIEP5_CTL	RWH	Device Control IN Endpoint x+1 Control Register
0xDE9C8	USB_DIEP5_INT	RWH	Device IN Endpoint x+1 Interrupt Register
0xDE9D0	USB_DIEP5_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register
0xDE9D4	USB_DIEP5_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
0xDE9D8	USB_DIEP5_DTXFSTS	R	Device IN Endpoint Transmit FIFO Status Register 1
0xDEB00	USB_DOEP0CTL	RWH	Device Control OUT Endpoint 0 Control Register
0xDEB08	USB_DOEP0INT	RW1	Device OUT Endpoint 0 Interrupt Register
0xDEB10	USB_DOEP0TSIZ	RW	Device OUT Endpoint 0 Transfer Size Register
0xDEB14	USB_DOEP0DMAADDR	RW	Device OUT Endpoint 0 DMA Address Register
0xDEB20	USB_DOEP0_CTL	RWH	Device Control OUT Endpoint x+1 Control Register
0xDEB28	USB_DOEP0_INT	RW	Device OUT Endpoint x+1 Interrupt Register
0xDEB30	USB_DOEP0_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
0xDEB34	USB_DOEP0_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register
...	USB_DOEPx_CTL	RWH	Device Control OUT Endpoint x+1 Control Register
...	USB_DOEPx_INT	RW	Device OUT Endpoint x+1 Interrupt Register
...	USB_DOEPx_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
...	USB_DOEPx_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register

Offset	Name	Type	Description
0xDEBC0	USB_DOEP5_CTL	RWH	Device Control OUT Endpoint x+1 Control Register
0xDEBC8	USB_DOEP5_INT	RW	Device OUT Endpoint x+1 Interrupt Register
0xDEBD0	USB_DOEP5_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
0xDEBD4	USB_DOEP5_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register
0xDEE00	USB_PCGCCTL	RWH	Power and Clock Gating Control Register
0xDF000	USB_FIFO0D0	RWH(a)	Device EP 0/Host Channel 0 FIFO
...	USB_FIFO0Dx	RWH(a)	Device EP 0/Host Channel 0 FIFO
0xDF7FC	USB_FIFO0D511	RWH(a)	Device EP 0/Host Channel 0 FIFO
0xE0000	USB_FIFO1D0	RWH(a)	Device EP 1/Host Channel 1 FIFO
...	USB_FIFO1Dx	RWH(a)	Device EP 1/Host Channel 1 FIFO
0xE07FC	USB_FIFO1D511	RWH(a)	Device EP 1/Host Channel 1 FIFO
0xE1000	USB_FIFO2D0	RWH(a)	Device EP 2/Host Channel 2 FIFO
...	USB_FIFO2Dx	RWH(a)	Device EP 2/Host Channel 2 FIFO
0xE17FC	USB_FIFO2D511	RWH(a)	Device EP 2/Host Channel 2 FIFO
0xE2000	USB_FIFO3D0	RWH(a)	Device EP 3/Host Channel 3 FIFO
...	USB_FIFO3Dx	RWH(a)	Device EP 3/Host Channel 3 FIFO
0xE27FC	USB_FIFO3D511	RWH(a)	Device EP 3/Host Channel 3 FIFO
0xE3000	USB_FIFO4D0	RWH(a)	Device EP 4/Host Channel 4 FIFO
...	USB_FIFO4Dx	RWH(a)	Device EP 4/Host Channel 4 FIFO
0xE37FC	USB_FIFO4D511	RWH(a)	Device EP 4/Host Channel 4 FIFO
0xE4000	USB_FIFO5D0	RWH(a)	Device EP 5/Host Channel 5 FIFO
...	USB_FIFO5Dx	RWH(a)	Device EP 5/Host Channel 5 FIFO
0xE47FC	USB_FIFO5D511	RWH(a)	Device EP 5/Host Channel 5 FIFO
0xE5000	USB_FIFO6D0	RWH(a)	Device EP 6/Host Channel 6 FIFO
...	USB_FIFO6Dx	RWH(a)	Device EP 6/Host Channel 6 FIFO
0xE57FC	USB_FIFO6D511	RWH(a)	Device EP 6/Host Channel 6 FIFO
0xE6000	USB_FIFO7D0	RWH(a)	Host Channel 7 FIFO
...	USB_FIFO7Dx	RWH(a)	Host Channel 7 FIFO
0xE67FC	USB_FIFO7D511	RWH(a)	Host Channel 7 FIFO

Offset	Name	Type	Description
0xE7000	USB_FIFO8D0	RWH(a)	Host Channel 8 FIFO
...	USB_FIFO8Dx	RWH(a)	Host Channel 8 FIFO
0xE77FC	USB_FIFO8D511	RWH(a)	Host Channel 8 FIFO
0xE8000	USB_FIFO9D0	RWH(a)	Host Channel 9 FIFO
...	USB_FIFO9Dx	RWH(a)	Host Channel 9 FIFO
0xE87FC	USB_FIFO9D511	RWH(a)	Host Channel 9 FIFO
0xE9000	USB_FIFO10D0	RWH(a)	Host Channel 10 FIFO
...	USB_FIFO10Dx	RWH(a)	Host Channel 10 FIFO
0xE97FC	USB_FIFO10D511	RWH(a)	Host Channel 10 FIFO
0xEA000	USB_FIFO11D0	RWH(a)	Host Channel 11 FIFO
...	USB_FIFO11Dx	RWH(a)	Host Channel 11 FIFO
0xEA7FC	USB_FIFO11D511	RWH(a)	Host Channel 11 FIFO
0xEB000	USB_FIFO12D0	RWH(a)	Host Channel 12 FIFO
...	USB_FIFO12Dx	RWH(a)	Host Channel 12 FIFO
0xEB7FC	USB_FIFO12D511	RWH(a)	Host Channel 12 FIFO
0xEC000	USB_FIFO13D0	RWH(a)	Host Channel 13 FIFO
...	USB_FIFO13Dx	RWH(a)	Host Channel 13 FIFO
0xEC7FC	USB_FIFO13D511	RWH(a)	Host Channel 13 FIFO
0xFE000	USB_FIFORAM0	RWH(a)	Direct Access to Data FIFO RAM for Debugging (2 KB)
...	USB_FIFORAMx	RWH(a)	Direct Access to Data FIFO RAM for Debugging (2 KB)
0xFE7FC	USB_FIFORAM511	RWH(a)	Direct Access to Data FIFO RAM for Debugging (2 KB)

38.7 Register Description

38.7.1 USB_CTRL - System 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	0	0	0x0		0	0	0													0			0		0		0x2		0			0
Access	RW	RW	RW		RW	RW	RW													RW			RW		RW		RW		RW			RW
Name	SDEN	PDEN	DCDEN		OTGPHYCTRLDIS	OTGIDINDIS	OTGCLKCDIS													IDCDEN			LEMIDLEEN		LEMPHYCTRL		LEMOSCCTRL		SELFPOWERED			VBUSENAP

Bit	Name	Reset	Access	Description
31	SDEN	0	RW	**Secondary Detection Enable**

This bit enables secondary detection (SD).

Value	Mode	Description
0	DISABLED	Disable secondary detection.
1	ENABLED	Enable secondary detection.

Bit	Name	Reset	Access	Description
30	PDEN	0	RW	**Primary Detection Enable**

This bit enables primary detection (PD).

Value	Mode	Description
0	DISABLED	Disable primary detection.
1	ENABLED	Enable primary detection.

Bit	Name	Reset	Access	Description
29:28	DCDEN	0x0	RW	**Data Contact Detection Enable**

This field enables and configures the data contact detection (DCD).

Value	Mode	Description
0	DISABLED	DCD is disabled.
2	TIMEOUT	Only DCD timeout will be initiated.
3	ENABLED	Full DCD operation (physical contact and timeout) will be initiated.

Bit	Name	Reset	Access	Description
27	OTGPHYCTRLDIS	0	RW	**OTG Control Signals to PHY Disable**

Set this bit to disable USB OTG control signals to USB PHY.

Value	Mode	Description
0	ENABLED	OTG control signals to PHY is enabled
1	DISABLED	OTG control signals to PHY is disabled

Bit	Name	Reset	Access	Description
26	OTGIDINDIS	0	RW	**OTG ID Input Disable**

Set this bit to disable the output of PHY ID schmitt trigger going to USB OTG. USB OTG ID input will be set to 1

Value	Mode	Description
0	ENABLED	PHY ID output is connected to OTG ID input
1	DISABLED	OTG ID input is set to 1

Bit	Name	Reset	Access	Description
25	OTGCLKCDIS	0	RW	**OTG CLKC Disable**

Set this bit to disable usb clkc to USB OTG core. clkc to USB AHB registers remains running

Value	Mode	Description
0	ENABLED	CLKC to OTG core is enabled
1	DISABLED	CLKC to OTG core is disabled

Bit	Name	Reset	Access	Description
24:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	IDCDEN	0	RW	**ID Pull-up Enable**

This bit enables ID pull-up current source.

Value	Mode	Description
0	DISABLED	Disable ACA detection.
1	ENABLED	Enable ACA detection.

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	LEMIDLEEN	0	RW	**Low Energy Mode on Bus Idle Enable**

Set this bit to enter low energy mode during bus idle.

Bit	Name	Reset	Access	Description
8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	LEMPHYCTRL	0	RW	**Low Energy Mode USB PHY Control**

Configuration for USB PHY control when Low Energy Mode is active

Value	Mode	Description
0	NONE	The USB PHY is not affected by Low Energy Mode.
1	LEM	The USB PHY is put into an energy saving state when Low Energy Mode is active.

Bit	Name	Reset	Access	Description
6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:4	LEMOSCCTRL	0x2	RW	**Low Energy Mode Oscillator Control**

Configuration for oscillator control when Low Energy Mode is active. LEMOSCCTRL should not be changed when LEM is enabled.

Value	Mode	Description
0	NONE	Low Energy Mode has no effect on neither USBC or USHFRCO.
1	GATE	The USBC clock is gated when Low Energy Mode is active.

Bit	Name	Reset	Access	Description
3	SELFPOWERED PHY Power	0	RW	**PHY Power**

Value	Mode	Description
0	LOW	Clear this bit to 0 to indicate vbus power setting. PHY will be enabled when ROUTE_PHYPEN is set and vbus is detected.
1	HIGH	Set this bit to 1 to indicate Self-power setting. PHY will then be enabled when ROUTE_PHYPEN is set.

2:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	VBUSENAP	0	RW	**VBUSEN Active Polarity**

Use this bit to select the active polarity of the USB_VBUSEN pin.

Value	Mode	Description
0	HIGH	USB_VBUSEN is active high.
1	LOW	USB_VBUSEN is active low.

38.7.2 USB_STATUS - System Status 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																	0		0	0	0	0	0	0						0		0
Access																	R		R	R	R	R	R	R						R		R
Name																	USBCDBUSY		ACALS	ACAFS	DCP	CDP	SDP	DCDTO						LEMACTIVE		VBUSDETH

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	USBCDBUSY	0	R	**USB Charger Detect Busy**
	This bit is set when USB charger detect recieved the start command until end of the enabled process.			
14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	ACALS	0	R	**ACA Low Speed TypeB Device**
	This bit is set when USB charger detect a Low Speed TypeB Device for ACA-A algorithm.			
12	ACAFS	0	R	**ACA Full Speed TypeB Device**
	This bit is set when USB charger detect a Full Speed TypeB Device for ACA-A algorithm.			
11	DCP	0	R	**Dedicated Charging Port Detected**
	This bit is set at the completion of a secondary detection phase if a Dedicated Charging Port has been detected.			
10	CDP	0	R	**Charging Downstream Port Detected**
	This bit is set at the completion of a secondary detection phase if a Charging Downstream Port has been detected.			
9	SDP	0	R	**Standard Downstream Port Detected**
	This bit is set at the completion of a primary detection phase if a Standard Downstream Port has been detected.			
8	DCDTO	0	R	**Data Contact Detection Timeout**
	This bit is set at the completion of a DCD operation if the operation was stopped due to DCD timeout.			

Value	Mode	Description
0	NOTIME	A DCD timeout was not triggered.
1	TIMEOUT	A DCD timeout was triggered.

7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	LEMACTIVE	0	R	**Low Energy Mode Active**
	This bit is set when Low Energy Mode is active.			
1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	VBUSDETH	0	R	**VBUS Detect High**
	VBUS is at logic high			

38.7.3 USB_IF - Interrupt Flag 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																					0	0	0	0							0	0
Access																					R	R	R	R							R	R
Name																					SD	PD	DCD	ERR							VBUSDETL	VBUSDETH

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	SD	0	R	**Secondary Detection Complete Interrupt Flag**
	This bit is set at the completion of a SD operation.			
10	PD	0	R	**Primary Detection Complete Interrupt Flag**
	This bit is set at the completion of a PD operation.			
9	DCD	0	R	**Data Contact Detection Complete Interrupt Flag**
	This bit is set at the completion of a DCD operation. The DCDTO bit will indicate whether the DCD operation timed out.			
8	ERR	0	R	**Detection Error Interrupt Flag**
	This bit indicates that an error occurred during the charger detect sequence. This bit will be set if the VBUS signal is disconnected while the charger detect circuit is alive.			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	VBUSDETL	0	R	**VBUS Detect Low Interrupt Flag**
	Set when VBUS transitions from logic HIGH to LOW.			
0	VBUSDETH	0	R	**VBUS Detect High Interrupt Flag**
	Set when VBUS transitions from logic LOW to HIGH.			

38.7.4 USB_IFS - Interrupt Flag Set 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																					0	0	0	0							0	0
Access																					W1	W1	W1	W1							W1	W1
Name																					SD	PD	DCD	ERR							VBUSDETL	VBUSDETH

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	SD	0	W1	**Set SD Interrupt Flag**
	Write 1 to set the SD interrupt flag			
10	PD	0	W1	**Set PD Interrupt Flag**
	Write 1 to set the PD interrupt flag			
9	DCD	0	W1	**Set DCD Interrupt Flag**
	Write 1 to set the DCD interrupt flag			
8	ERR	0	W1	**Set ERR Interrupt Flag**
	Write 1 to set the ERR interrupt flag			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	VBUSDETL	0	W1	**Set VBUSDETL Interrupt Flag**
	Write 1 to set the VBUSDETL interrupt flag			
0	VBUSDETH	0	W1	**Set VBUSDETH Interrupt Flag**
	Write 1 to set the VBUSDETH interrupt flag			

38.7.5 USB_IFC - Interrupt Flag Clear 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																					0	0	0	0							0	0
Access																					(R)W1	(R)W1	(R)W1	(R)W1							(R)W1	(R)W1
Name																					SD	PD	DCD	ERR							VBUSDETL	VBUSDETH

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	SD	0	(R)W1	**Clear SD Interrupt Flag**
	Write 1 to clear the SD interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
10	PD	0	(R)W1	**Clear PD Interrupt Flag**
	Write 1 to clear the PD interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
9	DCD	0	(R)W1	**Clear DCD Interrupt Flag**
	Write 1 to clear the DCD interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
8	ERR	0	(R)W1	**Clear ERR Interrupt Flag**
	Write 1 to clear the ERR interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	VBUSDETL	0	(R)W1	**Clear VBUSDETL Interrupt Flag**
	Write 1 to clear the VBUSDETL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	VBUSDETH	0	(R)W1	**Clear VBUSDETH Interrupt Flag**
	Write 1 to clear the VBUSDETH interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

38.7.6 USB_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																					0	0	0	0							0	0
Access																					RW	RW	RW	RW							RW	RW
Name																					SD	PD	DCD	ERR							VBUSDETL	VBUSDETH

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	SD	0	RW	**SD Interrupt Enable**
	Enable/disable the SD interrupt			
10	PD	0	RW	**PD Interrupt Enable**
	Enable/disable the PD interrupt			
9	DCD	0	RW	**DCD Interrupt Enable**
	Enable/disable the DCD interrupt			
8	ERR	0	RW	**ERR Interrupt Enable**
	Enable/disable the ERR interrupt			
7:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1	VBUSDETL	0	RW	**VBUSDETL Interrupt Enable**
	Enable/disable the VBUSDETL interrupt			
0	VBUSDETH	0	RW	**VBUSDETH Interrupt Enable**
	Enable/disable the VBUSDETH interrupt			

38.7.7 USB_ROUTE - I/O Routing 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																															0	0
Access																															RW	RW
Name																															VBUSENPEN	PHYPEN

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	VBUSENPEN	0	RW	**VBUSEN Pin Enable** When set, the USB_VBUSEN pin is enabled.
0	PHYPEN	0	RW	**USB PHY Pin Enable** When set, the USB PHY and USB pins are enabled. The USB_DP and USB_DM are changed from regular GPIO pins to USB pins.

38.7.8 USB_CDCONF - Charger Detect Configuration 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																							0x000								
Access																							RW								
Name																							DCDTOCONF								

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	DCDTOCONF	0x000	RW	**DCD Timeout (TDCD_TIMEOUT) Configuration** Use this field to adjust the TDCD_TIMEOUT (1ms steps) starting from 300ms.

38.7.9 USB_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																															0	0
Access																															W1	W1
Name																															STOPCD	STARTCD

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	STOPCD	0	W1	**Start Charger Detection in Progress**
	Write a 1 to this bit to stop detection process which is progress.			
0	STARTCD	0	W1	**Start Charger Detection Enabled**
	Write a 1 to this bit to start detection process which is enabled.			

38.7.10 USB_DATTRIM1 - Data TRIM 1 Values for USB DP and DM

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		0		0x24					
Access													RW		RW		RW		RW		RW		RW		RW		RW					
Name													TRDPFS		TFDPFS		TRDMFS		TFDMFS		VCRSFS		DLYPULLUPFS		ENDLYPULLUP		ROUT					

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*		
19:18	TRDPFS	0x0	RW	**Trim for DP Rise Time in FS**
	Used to match rise times in ~5% steps. 0: Min rise time; 3:Max rise time.			
17:16	TFDPFS	0x0	RW	**Trim for DP Fall Time in FS**
	Used to match fall times in ~5% steps. 0: Min fall time; 3:Max fall time.			
15:14	TRDMFS	0x0	RW	**Trim for DM Rise Time in FS**
	Used to match DM rise times in ~5% steps. 0: Min rise time; 3:Max rise time.			
13:12	TFDMFS	0x0	RW	**Trim for DM Fall Time in FS**
	Used to match DM fall times in ~5% steps. 0: Min fall time; 3:Max fall time.			
11:10	VCRSFS	0x0	RW	**Trim for Falling Crossover Voltage in FS**
	Apply for both DP and DM. 0: Min crossover voltage; 3:Max crossover voltage.			
9:8	DLYPULLUPFS	0x0	RW	**Trim for Rising Crossover Voltage in FS**
	Control rising crossover voltage delay of DP in FS and DM in LS. 0: Min crossover voltage; 3:Max crossover voltage.			
7	ENDLYPULLUP	0	RW	**Enables Delay of Pull in TX Mode for Both FS and LS**
	Enable delay of pull up in transmit mode to compensate for 1.5k pull up resistor. Full-speed: Delays pull up of DP. Low-speed: delays pull up of DM.			
6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	ROUT	0x24	RW	**Trim for DP and DM Output Impedance for Both FS and LS**
	[5:3] Control pull up resistance, [2:0] control pull down resistance; 0=min. resistance, 7=max resistance.			

38.7.11 USB_LEMCTRL - USB LEM 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																											0x067					
Access																											RW					
Name																											TIMEBASE					

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	TIMEBASE	0x067	RW	**Set the Number of LFC Clk Counts to Form 3ms**
		Only applicable to USB device mode with LEM enabled. Use this field to set the number of LFC clk (used by USB LEM) counts to form 3ms period. This period matches the idle period on usb device prior to entering suspend mode. This field should be programmed before enabling of LFC or LEM.		

38.7.12 USB_ROUTELOC0 - I/O Routing Location 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																											0x00					
Access																											RW					
Name																											VBUSENPENLOC					

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	VBUSENPENLOC	0x00	RW	**I/O Location**
		Decides the location of the VBUSENPEN pin.		

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

38.7.13 USB_GOTGCTL - OTG Control and Status Register

The OTG Control and Status register controls the behavior and reflects the status of the OTG function of the core.

Offset	Bit Position																													
0xDE000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0	0	0	0	0	1	0			0	0	0	0	0	0	0	0	0	0	0	0	0
Access											R	RW	R	R	R	R	RW			RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	R
Name											CURMOD	OTGVER	BSESVLD	ASESVLD	DBNCTIME	CONIDSTS	DBNCEFLTRBYPASS			EHEN	DEVHNPEN	HSTSETHNPEN	HNPREQ	HSTNEGSCS	BVALIDOVVAL	BVALIDOVEN	AVALIDOVVAL	AVALIDOVEN	VBVALIDOVVAL	VBVALIDOVEN	SESREQ	SESREQSCS

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	CURMOD	0	R	**Current Mode of Operation**
	Indicates the current mode. 1'b0: Device mode 1'b1: Host mode			
20	OTGVER	0	RW	**OTG Version**
	Indicates the OTG revision. 1'b0: OTG Version 1.3. In this version the core supports Data line pulsing and VBus pulsing for SRP. 1'b1: OTG Version 2.0. In this version the core supports only Data line pulsing for SRP.			
19	BSESVLD	0	R	**B-Session Valid**
	Indicates the Device mode transceiver status. 1'b0: B-session is not valid. 1'b1: B-session is valid. In OTG mode, you can use this bit to determine if the device is connected or disconnected. Note: If you do not enable OTG features (such as SRP and HNP), the read reset value will be 1.The vbus assigns the values internally for non- SRP or non-HNP configurations.			
18	ASESVLD	0	R	**A-Session Valid**
	Indicates the Host mode transceiver status. 1'b0: A-session is not valid 1'b1: A-session is valid Note: If you do not enabled OTG features (such as SRP and HNP), the read reset value will be 1.The vbus assigns the values internally for non- SRP or non-HNP configurations.			
17	DBNCTIME	0	R	**Long/Short Debounce Time**
	Indicates the debounce time of a detected connection. 1'b0: Long debounce time, used for physical connections (100 ms + 2.5 micro-sec) 1'b1: Short debounce time, used for soft connections (2.5 micro-sec)			
16	CONIDSTS	1	R	**Connector ID Status**
	Indicates the connector ID status on a connect event. 1'b0: The DWC_otg core is in A-Device mode 1'b1: The DWC_otg core is in B-Device mode			
15	DBNCEFLTRBY-PASS	0	RW	**Debounce Filter Bypass**
	Valid only when OTG signals Debounce Filters are selected in configuration Bypass Debounce filters for avalid, bvalid, vbusvalid, sessend, iddig signals when enabled. 1'b0: Disabled 1'b1: Enabled			
14:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	EHEN	0	RW	**Embedded Host Enable**
	1'b0: Disable Embedded Host Mode 1'b1: Enable Embedded Host Mode			

Bit	Name	Reset	Access	Description
11	DEVHNPEN	0	RW	**Device HNP Enabled**
	The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host. 1'b0: HNP is not enabled in the application 1'b1: HNP is enabled in the application			
10	HSTSETHNPEN	0	RW	**Host Set HNP Enable**
	The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device. 1'b0: Host Set HNP is not enabled 1'b1: Host Set HNP is enabled			
9	HNPREQ	0	RW	**HNP Request**
	The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HSTNEGSUCSTSCHNG) is SET. The core clears this bit when the HSTNEGSUCSTSCHNG bit is cleared. 1'b0: No HNP request 1'b1: HNP request			
8	HSTNEGSCS	0	R	**Host Negotiation Success**
	The core sets this bit when host negotiation is successful. The core clears this bit when the HNP Request (HNPREQ) bit in this register is SET. 1'b0: Host negotiation failure 1'b1: Host negotiation success			
7	BVALIDOVVAL	0	RW	**B-Peripheral Session Valid OverrideValue**
	This bit is used to set Override value for Bvalid signal when GOTGCTL.BVALIDOVEN is set. 1'b0 : Bvalid value is 1'b0 when GOTGCTL.BVALIDOVEN =1 1'b1 : Bvalid value is 1'b1 when GOTGCTL.BVALIDOVEN =1			
6	BVALIDOVEN	0	RW	**B-Peripheral Session Valid Override Enable**
	This bit is used to enable/disable the software to override the Bvalid signal using the GOTGCTL.BVALIDOVVAL. 1'b1 : Internally Bvalid received from the PHY is overridden with GOTGCTL.BVALIDOVVAL. 1'b0 : Override is disabled and bvalid signal from the respective PHY selected is used internally by the core.			
5	AVALIDOVVAL	0	RW	**A-Peripheral Session Valid OverrideValue**
	This bit is used to set Override value for Avalid signal when GOTGCTL.AVALIDOVEN is set. 1'b0 : Avalid value is 1'b0 when GOTGCTL.AVALIDOVEN =1 1'b1 : Avalid value is 1'b1 when GOTGCTL.AVALIDOVEN =1			
4	AVALIDOVEN	0	RW	**A-Peripheral Session Valid Override Enable**
	This bit is used to enable/disable the software to override the Avalid signal using the GOTGCTL.AVALIDOVVAL. 1'b1 : Internally Avalid received from the PHY is overridden with GOTGCTL.AVALIDOVVAL. 1'b0 : Override is disabled and avalid signal from the respective PHY selected is used internally by the core			
3	VBVALIDOVVAL	0	RW	**VBUS Valid OverrideValue**
	This bit is used to set Override value for vbusvalid signal when GOTGCTL.VBVALIDOVEN is set. 1'b0 : vbusvalid value is 1'b0 when GOTGCTL.VBVALIDOVEN =1 1'b1 : vbusvalid value is 1'b1 when GOTGCTL.VBVALIDOVEN =1			
2	VBVALIDOVEN	0	RW	**VBUS Valid Override Enable**
	This bit is used to enable/disable the software to override the Bvalid signal using the GOTGCTL.VBVALIDOVVAL. 1'b1 : Internally Bvalid received from the PHY is overridden with GOTGCTL.VBVALIDOVVAL. 1'b0 : Override is disabled and bvalid signal from the respective PHY selected is used internally by the core			
1	SESREQ	0	RW	**Session Request**
	The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HSTNEGSUCSTSCHNG) is SET. The core clears this bit when the HSTNEGSUCSTSCHNG bit is cleared. If you use the USB 1.1 Full-Speed Serial Transceiver interface to initiate the session request, the application must wait until the VBUS discharges to 0.2 V, after the B-Session Valid bit in this register (GOTGCTL.BSESVLD) is cleared. This discharge time varies between different PHYs and can be obtained from the PHY vendor. 1'b0: No session request 1'b1: Session request			
0	SESREQSCS	0	R	**Session Request Success**
	The core sets this bit when a session request initiation is successful. 1'b0: Session request failure 1'b1: Session request success			

38.7.14 USB_GOTGINT - OTG Interrupt Register

Offset	Bit Position																														
0xDE004	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset													0	0	0								0	0						0		
Access													RW	RW	RW								RW	RW						RW		
Name													DBNCEDONE	ADEVTOUTCHG	HSTNEGDET								HSTNEGSUCSTSCHNG	SESREQSUCSTSCHNG						SESENDDET		

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	DBNCEDONE	0	RW	**Debounce Done**
	The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is SET in the Core USB Configuration register (GUSBCFG.HNPCAP or GUSBCFG.SRPCAP, respectively).This bit can be set only by the core and the application should write 1 to clear it.			
18	ADEVTOUTCHG	0	RW	**A-Device Timeout Change**
	The core sets this bit to indicate that the A-device has timed out WHILE waiting FOR the B-device to connect.This bit can be set only by the core and the application should write 1 to clear it.			
17	HSTNEGDET	0	RW	**Host Negotiation Detected**
	The core sets this bit when it detects a host negotiation request on the USB.This bit can be set only by the core and the application should write 1 to clear it.			
16:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
9	HSTNEG-SUCSTSCHNG	0	RW	**Host Negotiation Success Status Change**
	The core sets this bit on the success or failure of a USB host negotiation request. The application must read the Host Negotiation Success bit of the OTG Control and Status register (GOTGCTL.HSTNEGSCS) to check For success or failure.This bit can be set only by the core and the application should write 1 to clear it.			
8	SESREQ-SUCSTSCHNG	0	RW	**Session Request Success Status Change**
	The core sets this bit on the success or failure of a session request. The application must read the Session Request Success bit in the OTG Control and Status register (GOTGCTL.SESREQSCS) to check For success or failure.This bit can be set only by the core and the application should write 1 to clear it.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	SESENDDET	0	RW	**Session End Detected**
	The core sets this bit when VBUS is in the range 0.8V - 2.0V. This bit can be set only by the core and the application should write 1 to clear it.			
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

38.7.15 USB_GAHBCFG - AHB Configuration Register

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

Offset	Bit Position																													
0xDE008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0	0	0													0	0		0	0x0				0
Access									RW	RW	RW													RW	RW		RW	RW				RW
Name									AHBSINGLE	NOTIALLDMAWRIT	REMMEMSUPP													PTXFEMPLVL	NPTXFEMPLVL		DMAEN	HBSTLEN				GLBLINTRMSK

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	AHBSINGLE	0	RW	**AHB Single Support**
	This bit when programmed supports Single transfers for the remaining data in a transfer when the DWC_otg core is operating in DMA mode. 1'b0: The remaining data in the transfer is sent using INCR burst size. 1'b1: The remaining data in the transfer is sent using Single burst size. Note: if this feature is enabled, the AHB RETRY and SPLIT transfers still have INCR burst type. Enable this feature when the AHB Slave connected to the DWC_otg core does not support INCR burst (and when Split, and Retry transactions are not being used in the bus).			
22	NOTIALLDMAWRIT	0	RW	**Notify All Dma Write Transactions**
	This bit is programmed to enable the System DMA Done functionality for all the DMA write Transactions corresponding to the Channel/Endpoint. This bit is valid only when GAHBCFG.REMMEMSUPP is set to 1. GAHBCFG.NOTIALLDMAWRIT = 1 - HSOTG core asserts int_dma_req for all the DMA write transactions on the AHB interface along with int_dma_done, chep_last_transact and chep_number signal informations. The core waits for sys_dma_done signal for all the DMA write transactions in order to complete the transfer of a particular Channel/Endpoint. GAHBCFG.NOTIALLDMAWRIT = 0 - HSOTG core asserts int_dma_req signal only for the last transaction of DMA write transfer corresponding to a particular Channel/Endpoint. Similarly, the core waits for sys_dma_done signal only for that transaction of DMA write to complete the transfer of a particular Channel/Endpoint.			
21	REMMEMSUPP	0	RW	**Remote Memory Support**
	This bit is programmed to enable the functionality to wait for the system DMA Done Signal for the DMA Write Transfers. GAHBCFG.REMMEMSUPP=1 - The int_dma_req output signal is asserted when HSOTG DMA starts write transfer to the external memory. When the core is done with the Transfers it asserts int_dma_done signal to flag the completion of DMA writes from HSOTG. The core then waits for sys_dma_done signal from the system to proceed further and complete the Data Transfer corresponding to a particular Channel/Endpoint. GAHBCFG.REMMEMSUPP=0 - The int_dma_req and int_dma_done signals are not asserted and the core proceeds with the assertion of the XferComp interrupt as soon as the DMA write transfer is done at the HSOTG Core Boundary and it doesn't wait for the sys_dma_done signal to complete the DATA transfers			
20:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	PTXFEMPLVL	0	RW	**Periodic TxFIFO Empty Level**
	Indicates when the Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register (GINTSTS.PTXFEMP) is triggered. This bit is used only in Slave mode.			

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	HALFEMPTY		USB_GINTSTS.PTXFEMP interrupt indicates that the Periodic TxFIFO is half empty.
	1	EMPTY		USB_GINTSTS.PTXFEMP interrupt indicates that the Periodic TxFIFO is completely empty.
7	NPTXFEMPLVL	0	RW	**Non-Periodic TxFIFO Empty Level**
	This bit is used only in Slave mode. In host mode this bit indicates when the Non-Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register (USB_GINTSTS.NPTXFEMP) is triggered. In device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (USB_DIEP0INT/USB_DIEPx_INT.TXFEMP) is triggered.			
	Value	Mode		Description
	0	HALFEMPTY		Host Mode: USB_GINTSTS.NPTXFEMP interrupt indicates that the Non-Periodic TxFIFO is half empty. Device Mode: USB_DIEP0INT/ USB_DIEPx_INT.TXFEMP interrupt indicates that the IN Endpoint TxFIFO is half empty.
	1	EMPTY		Host Mode: USB_GINTSTS.NPTXFEMP interrupt indicates that the Non-Periodic TxFIFO is completely empty. Device Mode: USB_DIEP0INT/USB_DIEPx_INT.TXFEMP interrupt indicates that the IN Endpoint TxFIFO is completely empty.
6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	DMAEN	0	RW	**DMA Enable**
	1'b0: Core operates in Slave mode 1'b1: Core operates in a DMA mode This bit is always 0 when Slave-Only mode has been selected			
4:1	HBSTLEN	0x0	RW	**Burst Length/Type**
	This field is used in DMA modes.			
	Value	Mode		Description
	0	SINGLE		Single transfer.
	1	INCR		Incrementing burst of unspecified length.
	3	INCR4		4-beat incrementing burst.
	5	INCR8		8-beat incrementing burst.
	7	INCR16		16-beat incrementing burst.
0	GLBLINTRMSK	0	RW	**Global Interrupt Mask**
	The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core. 1'b0: Mask the interrupt assertion to the application. 1'b1: Unmask the interrupt assertion to the application.			

38.7.16 USB_GUSBCFG - USB Configuration Register

This register can be used to configure the core after power-on or a changing to Host mode or Device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

Offset	Bit Position																													
0xDE00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0						0									0x5				0	0			0				0x0	
Access	W1	RW	RW	RW						RW									RW				RW	RW			RW				RW	
Name	CORRUPTTXPKT	FORCEDEVMODE	FORCEHSTMODE	TXENDDELAY						TERMSELDLPULSE									USBTRDTIM				HNPCAP	SRPCAP			FSINTF				TOUTCAL	

Bit	Name	Reset	Access	Description
31	CORRUPTTXPKT	0	W1	**Corrupt Tx packet (host and device)**
	This bit is for debug purposes only. Never Set this bit to 1. The application should always write 0 to this bit.			
30	FORCEDEVMODE	0	RW	**Force Device Mode**
	Writing a 1 to this bit forces the core to device mode irrespective of USB_ID input pin. 1'b0 : Normal Mode. 1'b1 : Force Device Mode. After setting the force bit, the application must wait at least 25 ms before the change to take effect.			
29	FORCEHSTMODE	0	RW	**Force Host Mode**
	Writing a 1 to this bit forces the core to host mode irrespective of USB_ID input pin. 1'b0 : Normal Mode. 1'b1 : Force Host Mode. After setting the force bit, the application must wait at least 25 ms before the change to take effect. When the simulation is in scale down mode, waiting for 500 micro sec is sufficient.			
28	TXENDDELAY	0	RW	**Tx End Delay**
	Writing 1'b1 to this bit enables the core to follow the TXENDDELAY timings as per UTMI+ specification 1.05 section 4.1.5 for opmode signal during remote wakeup. 1'b0 : Normal Mode. 1'b1 : Tx End delay.			
27:23	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
22	TERMSELDLPULSE	0	RW	**TermSel DLine Pulsing Selection**
	This bit selects utmi_termselect to drive data line pulse during SRP.			

Value	Mode	Description
0	TXVALID	Data line pulsing using utmi_txvalid.
1	TERMSEL	Data line pulsing using utmi_termsel.

21:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:10	USBTRDTIM	0x5	RW	**USB Turnaround Time**
	Sets the turnaround time in PHY clocks. Specifies the response time For a MAC request to the Packet FIFO Controller (PFC) to fetch data from the DFIFO (SPRAM). Always write this field to 5.			
9	HNPCAP	0	RW	**HNP-Capable**
	The application uses this bit to control the core's HNP capabilities. Set to enable HNP capability.			

Bit	Name	Reset	Access	Description
8	SRPCAP	0	RW	**SRP-Capable**
	The application uses this bit to control the core's SRP capabilities. If the core operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate VBUS and start a session. Set to enable SRP capability.			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	FSINTF	0	RW	**Full-Speed Serial Interface Select**
	Always write this bit to 0.			
4:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	TOUTCAL	0x0	RW	**Timeout Calibration (host and device)**
	Always write this field to 0.			

38.7.17 USB_GRSTCTL - Reset Register

The application uses this register to reset various hardware features inside the core.

Offset	Bit Position																													
0xDE010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	1	0																						0x00			0	0		0	0	0
Access	R	R																						RW			RW1	RW1		RW1	RW	RW1
Name	AHBIDLE	DMAREQ																						TXFNUM			TXFFLSH	RXFFLSH		FRMCNTRRST	PIUFSSFTRST	CSFTRST

Bit	Name	Reset	Access	Description
31	AHBIDLE	1	R	**AHB Master Idle**
	Indicates that the AHB Master State Machine is in the IDLE condition.			
30	DMAREQ	0	R	**DMA Request Signal**
	Indicates that the DMA request is in progress. Used For debug.			
29:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:6	TXFNUM	0x00	RW	**TxFIFO Number (host and device)**

This is the FIFO number that must be flushed using the TxFIFO Flush bit. This field must not be changed until the core clears the TxFIFO Flush bit.

Value	Mode	Description
0	F0	Host mode: Non-periodic TxFIFO flush. Device: Tx FIFO 0 flush
1	F1	Host mode: Periodic TxFIFO flush. Device: TXFIFO 1 flush.
2	F2	Device mode: TXFIFO 2 flush.
3	F3	Device mode: TXFIFO 3 flush.
4	F4	Device mode: TXFIFO 4 flush.
5	F5	Device mode: TXFIFO 5 flush.
6	F6	Device mode: TXFIFO 6 flush.
16	FALL	Flush all the transmit FIFOs in device or host mode.

5	TXFFLSH	0	RW1	**TxFIFO Flush**

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction. The application must write this bit only after checking that the core is neither writing to the TxFIFO nor reading from the TxFIFO. NAK Effective Interrupt ensures the core is not reading from the FIFO. USB_GRSTCTL.AHBIDLE ensures the core is not writing anything to the FIFO. Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear.

Bit	Name	Reset	Access	Description
4	RXFFLSH	0	RW1	**RxFIFO Flush**
	The application can flush the entire RxFIFO using this bit, but must first ensure that the core is not in the middle of a transaction. The application must only write to this bit after checking that the core is neither reading from the RxFIFO nor writing to the RxFIFO. The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks to clear.			
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	FRMCNTRRST	0	RW1	**Host Frame Counter Reset**
	The application writes this bit to reset the frame number counter inside the core. When the frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0. When application writes 1 to the bit, it might not be able to read back the value as it will get cleared by the core in a few clock cycles.			
1	PIUFSSFTRST	0	RW	**PIU FS Dedicated Controller Soft Reset**
	Resets the PIU FS Dedicated Controller All module state machines in FS Dedicated Controller of PIU are reset to the IDLE state. Used to reset the FS Dedicated controller in PIU in case of any PHY Errors like Loss of activity or Babble Error resulting in the PHY remaining in RX state for more than one frame boundary			
0	CSFTRST	0	RW1	**Core Soft Reset (host and device)**
	Resets the core by clearing the interrupts and all the CSR registers except the following register bits: USB_PCGCCTL.RSTPDWNMODULE, USB_PCGCCTL.GATEHCLK, USB_PCGCCTL.PWRCLMP, USB_GUSBCFG.FSINTF, USB_HCFG.FSLSPCLKSEL, USB_DCFG.DEVSPD. All module state machines (except the AHB Slave Unit) are reset to the IDLE state, and all the transmit FIFOs and the receive FIFO are flushed. Any transactions on the AHB Master are terminated as soon as possible, after gracefully completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit is cleared software must wait at least 3 clock cycles before doing any access to the core. Software must also must check that bit 31 of this register is 1 (AHB Master is IDLE) before starting any operation.			

38.7.18 USB_GINTSTS - Interrupt Register

This register interrupts the application for system-level events in the current mode (Device mode or Host mode). Some of the bits in this register are valid only in Host mode, while others are valid in Device mode only. This register also indicates the current mode. To clear the interrupt status bits of type RW1, the application must write 1 into the bit. The FIFO status interrupts are read only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically. The application

must clear the USB_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Offset	Bit Position																													
0xDE014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	1		1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0			0	0	1	0	0	0	0	0
Access	RW1	RW1	RW1	RW1		R	R	R	RW1	RW1	RW1	RW1	R	R	RW1		RW1	RW1	RW1	RW1	RW1	RW1			R	R	R	R	RW1	R	RW1	R
Name	WKUPINT	SESSREQINT	DISCONNINT	CONIDSTSCHNG		PTXFEMP	HCHINT	PRTINT	RESETDET	FETSUSP	INCOMPLP	INCOMPISOIN	OEPINT	IEPINT	EPMIS		EOPF	ISOOUTDROP	ENUMDONE	USBRST	USBSUSP	ERLYSUSP			GOUTNAKEFF	GINNAKEFF	NPTXFEMP	RXFLVL	SOF	OTGINT	MODEMIS	CURMOD

Bit	Name	Reset	Access	Description
31	WKUPINT	0	RW1	**Resume/Remote Wakeup Detected Interrupt (host and device)**
	Wakeup Interrupt during Suspend state. In Device mode this interrupt is asserted only when Host Initiated Resume is detected on USB. In Host mode this interrupt is asserted only when Device Initiated Remote Wakeup is detected on USB. This bit can be set only by the core and the application should write 1 to clear.			
30	SESSREQINT	0	RW1	**Session Request/New Session Detected Interrupt (host and device)**
	In Host mode, this interrupt is asserted when a session request is detected from the device. In Device mode, this interrupt is asserted when the VBUS voltage reaches the session-valid level. This bit can be set only by the core and the application should write 1 to clear.			
29	DISCONNINT	0	RW1	**Disconnect Detected Interrupt (host only)**
	Asserted when a device disconnect is detected. This bit can be set only by the core and the application should write 1 to clear it.			
28	CONIDSTSCHNG	1	RW1	**Connector ID Status Change (host and device)**
	The core sets this bit when there is a change in connector ID status. This bit can be set only by the core and the application should write 1 to clear it.			
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	PTXFEMP	1	R	**Periodic TxFIFO Empty (host only)**
	This interrupt is asserted when the Periodic Transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the Periodic Request Queue. The half or completely empty status is determined by the Periodic TxFIFO Empty Level bit in the Core AHB Configuration register (USB_GAHBCFG.PTXFEMPLVL).			
25	HCHINT	0	R	**Host Channels Interrupt (host only)**
	The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in Host mode). The application must read the Host All Channels Interrupt (USB_HAINT) register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding Host Channel-x Interrupt (USB_HCx_INT) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the USB_HCx_INT register to clear this bit.			
24	PRTINT	0	R	**Host Port Interrupt (host only)**
	The core sets this bit to indicate a change in port status in Host mode. The application must read the Host Port Control and Status (USB_HPRT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the Host Port Control and Status register to clear this bit.			

Bit	Name	Reset	Access	Description
23	RESETDET	0	RW1	**Reset detected Interrupt (device only)**
	In Device mode, this interrupt is asserted when a reset is detected on the USB in EM2 when the device is in Suspend. In Host mode, this interrupt is not asserted.			
22	FETSUSP	0	RW1	**Data Fetch Suspended (device only)**
	This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or Request Queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application: Sets a Global non-periodic IN NAK handshake, Disables In endpoints, Flushes the FIFO, Determines the token sequence from the IN Token Sequence, Re-enables the endpoints, Clears the Global non-periodic IN NAK handshake. If the Global non-periodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an IN Token Received when FIFO Empty interrupt. The OTG then sends the host a NAK response. To avoid this scenario, the application can check the USB_GINTSTS.FETSUSP interrupt, which ensures that the FIFO is full before clearing a Global NAK handshake. Alternatively, the application can mask the IN Token Received when FIFO Empty interrupt when clearing a Global IN NAK handshake.			
21	INCOMPLP	0	RW1	**Incomplete Periodic Transfer (device only)**
	In Host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending which are scheduled for the current frame. In Device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This bit can be set only by the core and the application should write 1 to clear it.			
20	INCOMPISOIN	0	RW1	**Incomplete Isochronous IN Transfer (device only)**
	The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame.			
19	OEPINT	0	R	**OUT Endpoints Interrupt (device only)**
	The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (USB_DAINT) register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding Device OUT Endpoint-x Interrupt (USB_DOEP0INT/USB_DOEPx_INT) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding USB_DOEP0INT/USB_DOEPx_INT register to clear this bit.			
18	IEPINT	0	R	**IN Endpoints Interrupt (device only)**
	The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (USB_DAINT) register to determine the exact number of the IN endpoint on Device IN Endpoint-x Interrupt (USB_DIEP0INT/USB_DIEPx_INT) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding USB_DIEP0INT/USB_DIEPx_INT register to clear this bit.			
17	EPMIS	0	RW1	**Endpoint Mismatch Interrupt (device only)**
	Note: This interrupt is valid only in shared FIFO operation. Indicates that an IN token has been received For a non-periodic endpoint, but the data for another endpoint is present in the top of the Non-periodic Transmit FIFO and the IN endpoint mismatch count programmed by the application has expired.			
16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	EOPF	0	RW1	**End of Periodic Frame Interrupt**
	Indicates that the period specified in the Periodic Frame Interval field of the Device Configuration register (DCFG.PERFRINT) has been reached in the current frame.			
14	ISOOUTDROP	0	RW1	**Isochronous OUT Packet Dropped Interrupt (device only)**
	The core sets this bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO does not have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint.			
13	ENUMDONE	0	RW1	**Enumeration Done (device only)**
	The core sets this bit to indicate that speed enumeration is complete. The application must read the Device Status (USB_DSTS) register to obtain the enumerated speed.			

Bit	Name	Reset	Access	Description
12	USBRST	0	RW1	**USB Reset (device only)**
	The core sets this bit to indicate that a reset is detected on the USB.			
11	USBSUSP	0	RW1	**USB Suspend (device only)**
	The core sets this bit to indicate that a suspend was detected on the USB. The core enters the Suspended state when there is no activity on the bus for an extended period of time.			
10	ERLYSUSP	0	RW1	**Early Suspend (device only)**
	The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.			
9:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	GOUTNAKEFF	0	R	**Global OUT NAK Effective (device only)**
	Indicates that the Set Global OUT NAK bit in the Device Control register (USB_DCTL.SGOUTNAK), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear Global OUT NAK bit in the Device Control register (USB_DCTL.CGOUTNAK).			
6	GINNAKEFF	0	R	**Global IN Non-periodic NAK Effective (device only)**
	Indicates that the Set Global Non-periodic IN NAK bit in the Device Control register (USB_DCTL.SGNPINNAK), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear Global Non-periodic IN NAK bit in the Device Control register (USB_DCTL.CGNPINNAK). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.			
5	NPTXFEMP	1	R	**Non-Periodic TxFIFO Empty (host only)**
	This interrupt is asserted when the Non-periodic TxFIFO is either half or completely empty, and there is space for at least one entry to be written to the Non-periodic Transmit Request Queue. The half or completely empty status is determined by the Non-periodic TxFIFO Empty Level bit in the Core AHB Configuration register (USB_GAHBCFG.NPTXFEMPLVL).			
4	RXFLVL	0	R	**RxFIFO Non-Empty (host and device)**
	Indicates that there is at least one packet pending to be read from the RxFIFO.			
3	SOF	0	RW1	**Start of Frame (host and device)**
	In Host mode, the core sets this bit to indicate that an SOF (FS) or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt. In Device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the Device Status register to get the current frame number. This interrupt is seen only when the core is operating at full-speed (FS). This bit can be set only by the core and the application should write 1 to clear it.			
2	OTGINT	0	R	**OTG Interrupt (host and device)**
	The core sets this bit to indicate an OTG protocol event. The application must read the OTG Interrupt Status (USB_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the USB_GOTGINT register to clear this bit.			
1	MODEMIS	0	RW1	**Mode Mismatch Interrupt (host and device)**
	The core sets this bit when the application is trying to access a Host mode register, when the core is operating in Device mode or when the application accesses a Device mode register, when the core is operating in Host mode. The register access is ignored by the core internally and does not affect the operation of the core. This bit can be set only by the core and the application should write 1 to clear it.			
0	CURMOD	0	R	**Current Mode of Operation (host and device)**
	Indicates the current mode.			

Value	Mode	Description
0	DEVICE	Device mode.
1	HOST	Host mode.

Bit	Name	Reset	Access	Description

38.7.19 USB_GINTMSK - Interrupt Mask Register

This register works with the Interrupt Register (USB_GINTSTS) to interrupt the application. When an interrupt bit is masked (bit is 0), the interrupt associated with that bit is not generated. However, the USB_GINTSTS register bit corresponding to that interrupt is still set.

Offset	Bit Position																														
0xDE018	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	
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	
Name	WKUPINTMSK	SESSREQINTMSK	DISCONNINTMSK	CONIDSTSCHNGMSK		PTXFEMPMSK	HCHINTMSK	PRTINTMSK	RESETDETMSK	FETSUSPMSK	INCOMPLPMSK	INCOMPISOINMSK	OEPINTMSK	IEPINTMSK	EPMISMSK		EOPFMSK	ISOOUTDROPMSK	ENUMDONEMSK	USBRSTMSK	USBSUSPMSK	ERLYSUSPMSK			GOUTNAKEFFMSK	GINNAKEFFMSK	NPTXFEMPMSK	RXFLVLMSK	SOFMSK	OTGINTMSK	MODEMISMSK	

Bit	Name	Reset	Access	Description
31	WKUPINTMSK	0	RW	**Resume/Remote Wakeup Detected Interrupt Mask (host and device)**
	Set to 1 to unmask WKUPINT interrupt.			
30	SESSREQINTMSK	0	RW	**Session Request/New Session Detected Interrupt Mask (host and device)**
	Set to 1 to unmask SESSREQINT interrupt.			
29	DISCONNINTMSK	0	RW	**Disconnect Detected Interrupt Mask (host and device)**
	Set to 1 to unmask DISCONNINT interrupt.			
28	CONIDSTSCHNGMSK	0	RW	**Connector ID Status Change Mask (host and device)**
	Set to 1 to unmask CONIDSTSCHNG interrupt.			
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	PTXFEMPMSK	0	RW	**Periodic TxFIFO Empty Mask (host only)**
	Set to 1 to unmask PTXFEMP interrupt.			
25	HCHINTMSK	0	RW	**Host Channels Interrupt Mask (host only)**
	Set to 1 to unmask HCHINT interrupt.			
24	PRTINTMSK	0	RW	**Host Port Interrupt Mask (host only)**
	Set to 1 to unmask PRTINT interrupt.			
23	RESETDETMSK	0	RW	**Reset detected Interrupt Mask (device only)**
	Set to 1 to unmask RESETDET interrupt.			
22	FETSUSPMSK	0	RW	**Data Fetch Suspended Mask (device only)**
	Set to 1 to unmask FETSUSP interrupt.			
21	INCOMPLPMSK	0	RW	**Incomplete Periodic Transfer Mask (host only)**
	Set to 1 to unmask INCOMPLP interrupt.			

Bit	Name	Reset	Access	Description
20	INCOMPISOINMSK	0	RW	**Incomplete Isochronous IN Transfer Mask (device only)**
	Set to 1 to unmask INCOMPISOIN interrupt.			
19	OEPINTMSK	0	RW	**OUT Endpoints Interrupt Mask (device only)**
	Set to 1 to unmask OEPINT interrupt.			
18	IEPINTMSK	0	RW	**IN Endpoints Interrupt Mask (device only)**
	Set to 1 to unmask IEPINT interrupt.			
17	EPMISMSK	0	RW	**Endpoint Mismatch Interrupt Mask (device only)**
	Set to 1 to unmask EPMIS interrupt.			
16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	EOPFMSK	0	RW	**End of Periodic Frame Interrupt Mask (device only)**
	Set to 1 to unmask EOPF interrupt.			
14	ISOOUTDROPMSK	0	RW	**Isochronous OUT Packet Dropped Interrupt Mask (device only)**
	Set to 1 to unmask ISOOUTDROP interrupt.			
13	ENUMDONEMSK	0	RW	**Enumeration Done Mask (device only)**
	Set to 1 to unmask ENUMDONE interrupt.			
12	USBRSTMSK	0	RW	**USB Reset Mask (device only)**
	Set to 1 to unmask USBRST interrupt.			
11	USBSUSPMSK	0	RW	**USB Suspend Mask (device only)**
	Set to 1 to unmask USBSUSP interrupt.			
10	ERLYSUSPMSK	0	RW	**Early Suspend Mask (device only)**
	Set to 1 to unmask ERLYSUSP interrupt.			
9:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	GOUTNAKEFFMSK	0	RW	**Global OUT NAK Effective Mask (device only)**
	Set to 1 to unmask GOUTNAKEFF interrupt.			
6	GINNAKEFFMSK	0	RW	**Global Non-periodic IN NAK Effective Mask (device only)**
	Set to 1 to unmask GINNAKEFF interrupt.			
5	NPTXFEMPMSK	0	RW	**Non-Periodic TxFIFO Empty Mask (host only)**
	Set to 1 to unmask NPTXFEMP interrupt.			
4	RXFLVLMSK	0	RW	**Receive FIFO Non-Empty Mask (host and device)**
	Set to 1 to unmask RXFLVL interrupt.			
3	SOFMSK	0	RW	**Start of Frame Mask (host and device)**
	Set to 1 to unmask SOF interrupt.			
2	OTGINTMSK	0	RW	**OTG Interrupt Mask (host and device)**
	Set to 1 to unmask OTGINT interrupt.			
1	MODEMISMSK	0	RW	**Mode Mismatch Interrupt Mask (host and device)**
	Set to 1 to unmask MODEMIS interrupt.			

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

38.7.20 USB_GRXSTSR - Receive Status Debug Read Register

A read to the Receive Status Debug Read register returns the contents of the top of the Receive FIFO. The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status pop/read when the receive FIFO is empty

and returns a value of 0x00000000. The application must only pop the Receive Status FIFO when the Receive FIFO Non-Empty bit of the Core Interrupt register (USB_GINTSTS.RXFLVL) is asserted.

Offset	Bit Position																														
0xDE01C	31	30	29	28	27	26	25	24	23	22	21	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		0x000											0x0			
Access								R				R				R		R											R			
Name								FN				PKTSTS				DPID		BCNT											CHNUM			

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*		
24:21	FN	0x0	R	**Frame Number**
	This is the least significant 4 bits of the Frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.			
20:17	PKTSTS	0x0	R	**Packet Status**
	Indicates the status of the received packet.			

Value	Mode	Description
1	GOUTNAK	Device mode: Global OUT NAK (triggers an interrupt).
2	PKTRCV	Host mode: IN data packet received. Device mode: OUT data packet received.
3	XFERCOMPL	Host mode: IN transfer completed (triggers an interrupt). Device mode: OUT transfer completed (triggers an interrupt).
4	SETUPCOMPL	Device mode: SETUP transaction completed (triggers an interrupt).
5	TGLERR	Host mode: Data toggle error (triggers an interrupt).
6	SETUPRCV	Device mode: SETUP data packet received.
7	CHLT	Host mode: Channel halted (triggers an interrupt).

16:15	DPID	0x0	R	**Data PID**
	Host mode: Indicates the Data PID of the received packet. Device mode: Indicates the Data PID of the received OUT data packet.			

Value	Mode	Description
0	DATA0	DATA0 PID.
1	DATA1	DATA1 PID.
2	DATA2	DATA2 PID.
3	MDATA	MDATA PID.

14:4	BCNT	0x000	R	**Byte Count**
	Indicates the byte count of the received IN data packet. Mode: Device only Byte Count (BCNT) Indicates the byte count of the received data packet.			

Bit	Name	Reset	Access	Description
3:0	CHNUM	0x0	R	**Channel Number** Indicates the channel number to which the current received packet belongs. Mode: Device only Endpoint Number (EPNUM) Indicates the endpoint number to which the current received packet belongs.

38.7.21 USB_GRXSTSP - Receive Status Read /Pop Register

A read to the Receive Status Read and Pop register returns the contents of the top of the Receive FIFO and pops the top data entry out of the RxFIFO. The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive

status pop/read when the receive FIFO is empty and returns a value of 0x00000000. The application must only pop the Receive Status FIFO when the Receive FIFO Non-Empty bit of the Core Interrupt register (USB_GINTSTS.RXFLVL) is asserted.

Offset	Bit Position																														
0xDE020	31	30	29	28	27	26	25	24	23	22	21	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		0x000											0x0			
Access								R				R				R		R											R			
Name								FN				PKTSTS				DPID		BCNT											CHNUM			

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*		
24:21	FN	0x0	R	**Frame Number**
	This is the least significant 4 bits of the Frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.			
20:17	PKTSTS	0x0	R	**Packet Status (host or device)**
	Indicates the status of the received packet			

Value	Mode	Description
1	GOUTNAK	Device mode: Global OUT NAK (triggers an interrupt).
2	PKTRCV	Host mode: IN data packet received. Device mode: OUT data packet received.
3	XFERCOMPL	Host mode: IN transfer completed (triggers an interrupt). Device mode: OUT transfer completed (triggers an interrupt).
4	SETUPCOMPL	Device mode: SETUP transaction completed (triggers an interrupt).
5	TGLERR	Host mode: Data toggle error (triggers an interrupt).
6	SETUPRCV	Device mode: SETUP data packet received.
7	CHLT	Host mode: Channel halted (triggers an interrupt).

16:15	DPID	0x0	R	**Data PID (host or device)**
	Host mode: Indicates the Data PID of the received packet. Device mode: Indicates the Data PID of the received OUT data packet.			

Value	Mode	Description
0	DATA0	DATA0 PID.
1	DATA1	DATA1 PID.
2	DATA2	DATA2 PID.
3	MDATA	MDATA PID.

14:4	BCNT	0x000	R	**Byte Count**
	Indicates the byte count of the received IN data packet. Mode: Device only Byte Count (BCNT) Indicates the byte count of the received data packet.			

Bit	Name	Reset	Access	Description
3:0	CHNUM	0x0	R	**Channel Number**
	Indicates the channel number to which the current received packet belongs. Mode: Device only Endpoint Number (EPNUM) Indicates the endpoint number to which the current received packet belongs.			

38.7.22 USB_GRXFSIZ - Receive FIFO Size Register

The application can program the RAM size that must be allocated to the RxFIFO.

Offset	Bit Position																															
0xDE024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x200					
Access																											RW					
Name																											RXFDEP					

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	RXFDEP	0x200	RW	**RxFIFO Depth**
	This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.			

38.7.23 USB_GNPTXFSIZ - Non-periodic Transmit FIFO Size Register

The application can program the RAM size and the memory start address for the Non-periodic TxFIFO.

Offset	Bit Position																													
0xDE028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0200																0x0200															
Access	RW																RW															
Name	NPTXFINEPTXF0DEP																NPTXFSTADDR															

Bit	Name	Reset	Access	Description
31:16	NPTXFI-NEPTXF0DEP	0x0200	RW	**Non-periodic TxFIFO Depth (host only) / IN Endpoint TxFIFO 0 Depth (device only)**
	This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.			
15:0	NPTXFSTADDR	0x0200	RW	**Non-periodic Transmit RAM Start Address**
	This field contains the memory start address for Non-periodic Transmit FIFO RAM. Programmed values must not exceed the reset value.			

38.7.24 USB_GNPTXSTS - Non-periodic Transmit FIFO/Queue Status Register

This register is used in host mode only. This read-only register contains the free space information for the Non-periodic TxFIFO and the Nonperiodic Transmit Request Queue.

Offset	Bit Position																													
0xDE02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x00							0x08								0x0200															
Access		R							R								R															
Name		NPTXQTOP							NPTXQSPCAVAIL								NPTXFSPCAVAIL															

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:24	NPTXQTOP	0x00	R	**Top of the Non-periodic Transmit Request Queue** Entry in the Non-periodic Tx Request Queue that is currently being processed by the MAC. Bits [6:3]: Channel/endpoint number. Bits [2:1]: 00: IN/OUT token, 01: Zero-length transmit packet (device IN/host OUT), 10: Unused, 11: Channel halt command. Bit [0]: Terminate (last Entry for selected channel/endpoint).
23:16	NPTXQSPCAVAIL	0x08	R	**Non-periodic Transmit Request Queue Space Available** Indicates the amount of free space (locations) available in the Non-periodic Transmit Request Queue. This queue holds both IN and OUT requests in Host mode. Device mode has only IN requests.
15:0	NPTXFSPCAVAIL	0x0200	R	**Non-periodic TxFIFO Space Avail** Indicates the amount of free space available in the Non-periodic TxFIFO. Values are in terms of 32-bit words.

38.7.25 USB_GSNPSID - Synopsys ID Register

Offset	Bit Position																													
0xDE040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x4F54330A																															
Access	R																															
Name	SYNOPSYSID																															

Bit	Name	Reset	Access	Description
31:0	SYNOPSYSID	0x4F54330A	R	
	Release number of the DWC_otg core being used currently OTG			

38.7.26 USB_GDFIFOCFG - Global DFIFO Configuration Register

Offset	Bit Position																													
0xDE05C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x01F2																0x0200															
Access	RW																RW															
Name	EPINFOBASEADDR																GDFIFOCFG															

Bit	Name	Reset	Access	Description
31:16	EPINFOBASEADDR	0x01F2	RW	
	This field provides the start address of the EP info controller.			
15:0	GDFIFOCFG	0x0200	RW	
	This field is for dynamic programming of the DFIFO Size. This value takes effect only when the application programs a non zero value to this register. The core does not have any corrective logic if the FIFO sizes are programmed incorrectly.			

38.7.27 USB_HPTXFSIZ - Host Periodic Transmit FIFO Size Register

This register holds the size and the memory start address of the Periodic TxFIFO.

Offset	Bit Position																													
0xDE100	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x200															0x400				
Access												RW															RW				
Name												PTXFSIZE															PTXFSTADDR				

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*		
25:16	PTXFSIZE	0x200	RW	**Host Periodic TxFIFO Depth**
	This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.			
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:0	PTXFSTADDR	0x400	RW	**Host Periodic TxFIFO Start Address**
	This field contains the memory start address for Host Periodic TxFIFO.			

38.7.28 USB_DIEPTXF1 - Device IN Endpoint Transmit FIFO Size Register 1

This register holds the size and memory start address of IN endpoint TxFIFO 1 in Device mode. For IN endpoint FIFO 0 use USB_GNPTXFSIZ register for programming the size and memory start address.

Offset	Bit Position																													
0xDE104	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x200															0x400				
Access												RW															RW				
Name												INEPNTXFDEP															INEPNTXFSTADDR				

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*		
25:16	INEPNTXFDEP	0x200	RW	**IN Endpoint TxFIFO Depth** This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:0	INEPNTXFSTADDR	0x400	RW	**IN Endpoint FIFOn Transmit RAM Start Address** This field contains the memory start address For IN endpoint Transmit FIFO 1.

38.7.29 USB_DIEPTXF2 - Device IN Endpoint Transmit FIFO Size Register 2

Offset	Bit Position																													
0xDE108	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x200																0x600					
Access											RW																RW					
Name											INEPNTXFDEP																INEPNTXFSTADDR					

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*		
25:16	INEPNTXFDEP	0x200	RW	**IN Endpoint TxFIFO Depth** This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.
15:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:0	INEPNTXFSTADDR	0x600	RW	**IN Endpoint FIFOn Transmit RAM Start Address** This field contains the memory start address for IN endpoint Transmit FIFO 2.

38.7.30 USB_DIEPTXF3 - Device IN Endpoint Transmit FIFO Size Register 3

Offset	Bit Position																													
0xDE10C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x200															0x800						
Access											RW															RW						
Name											INEPNTXFDEP															INEPNTXFSTADDR						

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*		
25:16	INEPNTXFDEP	0x200	RW	**IN Endpoint TxFIFO Depth** This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	INEPNTXFSTADDR	0x800	RW	**IN Endpoint FIFOn Transmit RAM Start Address** This field contains the memory start address for IN endpoint Transmit FIFO 3.

38.7.31 USB_DIEPTXF4 - Device IN Endpoint Transmit FIFO Size Register 4

Offset	Bit Position																													
0xDE110	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x200																0xA00				
Access											RW																RW				
Name											INEPNTXFDEP																INEPNTXFSTADDR				

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*		
25:16	INEPNTXFDEP	0x200	RW	**IN Endpoint TxFIFO Depth** This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	INEPNTXFSTADDR	0xA00	RW	**IN Endpoint FIFOn Transmit RAM Start Address** This field contains the memory start address for IN endpoint Transmit FIFO 4.

38.7.32 USB_DIEPTXF5 - Device IN Endpoint Transmit FIFO Size Register 5

Offset	Bit Position																													
0xDE114	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x200																0xC00				
Access											RW																RW				
Name											INEPNTXFDEP																INEPNTXFSTADDR				

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*		
25:16	INEPNTXFDEP	0x200	RW	**IN Endpoint TxFIFO Depth** This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	INEPNTXFSTADDR	0xC00	RW	**IN Endpoint FIFOn Transmit RAM Start Address** This field contains the memory start address for IN endpoint Transmit FIFO 5.

38.7.33 USB_DIEPTXF6 - Device IN Endpoint Transmit FIFO Size Register 6

Offset	Bit Position																													
0xDE118	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x200															0xE00				
Access												RW															RW				
Name												INEPNTXFDEP															INEPNTXFSTADDR				

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*		
25:16	INEPNTXFDEP	0x200	RW	**IN Endpoint TxFIFO Depth** This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:0	INEPNTXFSTADDR	0xE00	RW	**IN Endpoint FIFOn Transmit RAM Start Address** This field contains the memory start address for IN endpoint Transmit FIFO 6.

38.7.34 USB_HCFG - Host Configuration Register

This register configures the core after power-on. Do not make changes to this register after initializing the host.

Offset	Bit Position																													
0xDE400	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0																0x02								0					0	0x0	
Access	RW																RW								RW					RW	RW	
Name	MODECHTIMEN																RESVALID								ENA32KHZS					FSLSSUPP	FSLSPCLKSEL	

Bit	Name	Reset	Access	Description
31	MODECHTIMEN	0	RW	**Mode Change Time**
	This bit is used to enable/disable the Host core to wait 200 clock cycles at the end of Resume before changing the PHY opmode to normal operation. When set to 0 the Host core waits for either 200 PHY clock cycles or a linestate of SE0 at the end of resume to the change the PHY opmode to normal operation. When set to 1 the Host core waits only for a linstate of SE0 at the end of resume to change the PHY opmode to normal operation.			
30:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15:8	RESVALID	0x02	RW	**Resume Validation Period**
	This field is effective only when HCFG.ENA32KHZS is set. It will control the resume period when the core resumes from suspend. The core counts for 'RESVALID' number of clock cycles to detect a valid resume when this is set.			
7	ENA32KHZS	0	RW	**Enable 32 kHz Suspend Mode**
	When FS PHY interface is chosen and this bit is set, the core expects that the USB Core Clock during Suspend is switched from 48 MHz to 32 KHz.			
6:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	FSLSSUPP	0	RW	**FS- and LS-Only Support**
	The application uses this bit to control the core's enumeration speed. Using this bit, the application can make the core enumerate as a FS host, even If the connected device supports HS traffic. Do not make changes to this field after initial programming.			

Value	Mode	Description
0	HSFSLS	HS/FS/LS, based on the maximum speed supported by the connected device.
1	FSLS	FS/LS-only, even If the connected device can support HS.

1:0	FSLSPCLKSEL	0x0	RW	**FS/LS PHY Clock Select**
	Use this field to set the internal PHY clock frequency. Set to 48 MHz in FS Host mode and 6 MHz in LS Host mode. When you select a 6 MHz clock during LS mode, you must do a soft reset.			

Value	Mode	Description
1	DIV1	Internal PHY clock is running at 48 MHz (undivided).
2	DIV8	Internal PHY clock is running at 6 MHz (48 MHz divided by 8).

38.7.35 USB_HFIR - Host Frame Interval Register

This register stores the frame interval information for the current speed to which the core has enumerated.

Offset	Bit Position																													
0xDE404	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0	0xEA60															
Access																RW	RW															
Name																HFIRRLDCTRL	FRINT															

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	HFIRRLDCTRL	0	RW	**Reload Control**

This bit allows dynamic reloading of the HFIR register during run time. This bit needs to be programmed during initial configuration and its value should not be changed during runtime.

Value	Mode	Description
0	STATIC	The HFIR cannot be reloaded dynamically.
1	DYNAMIC	The HFIR can be dynamically reloaded during runtime.

Bit	Name	Reset	Access	Description
15:0	FRINT	0xEA60	RW	**Frame Interval**

The value that the application programs to this field specifies the interval between two consecutive SOFs (FS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the Port Enable bit of the Host Port Control and Status register (USB_HPRT.PRTENA) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY Clock Select field of the Host Configuration register (USB_HCFG.FSLSPCLKSEL). Do not change the value of this field after the initial configuration. Set to 48000 (1 ms at 48 MHz) for FS and 6000 (1 ms at 6 MHz) for LS.

38.7.36 USB_HFNUM - Host Frame Number/Frame Time Remaining Register

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

Offset	Bit Position																													
0xDE408	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0000																0x3FFF															
Access	R																R															
Name	FRREM																FRNUM															

Bit	Name	Reset	Access	Description
31:16	FRREM	0x0000	R	**Frame Time Remaining** Indicates the amount of time remaining in the current Frame (FS/LS), in terms of USB Core Clocks. This field decrements on each USB Core Clock. When it reaches zero, this field is reloaded with the value in the Frame Interval register and a new SOF is transmitted on the USB.
15:0	FRNUM	0x3FFF	R	**Frame Number** This field increments when a new SOF is transmitted on the USB, and is reset to 0 when it reaches 16'h3FFF.

38.7.37 USB_HPTXSTS - Host Periodic Transmit FIFO/Queue Status Register

This read-only register contains the free space information for the Periodic TxFIFO and the Periodic Transmit Request Queue.

Offset	Bit Position																													
0xDE410	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00								0x08								0x0200															
Access	R								R								R														
Name	PTXQTOP								PTXQSPCAVAIL								PTXFSPCAVAIL															

Bit	Name	Reset	Access	Description
31:24	PTXQTOP	0x00	R	**Top of the Periodic Transmit Request Queue**
	This indicates the Entry in the Periodic Tx Request Queue that is currently being processes by the MAC. This register is used for debugging. Bit [7]: Odd/Even Frame. 0: send in even Frame, 1: send in odd Frame. Bits [6:3]: Channel/endpoint number. Bits [2:1]: Type. 00: IN/OUT, 01: Zero-length packet, 10: Unused, 11: Disable channel command. Bit [0]: Terminate (last Entry for the selected channel/endpoint).			
23:16	PTXQSPCAVAIL	0x08	R	**Periodic Transmit Request Queue Space Available**
	Indicates the number of free locations available to be written in the Periodic Transmit Request Queue. This queue holds both IN and OUT requests.			
15:0	PTXFSPCAVAIL	0x0200	R	**Periodic Transmit Data FIFO Space Available**
	Indicates the number of free locations available to be written to in the Periodic TxFIFO. Values are in terms of 32-bit words.			

38.7.38 USB_HAINT - Host All Channels Interrupt Register

When a significant event occurs on a channel, the Host All Channels Interrupt register interrupts the application using the Host Channels Interrupt bit of the Core Interrupt register (USB_GINTSTS.HCHINT). There is one interrupt bit per channel. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Host Channel x Interrupt register.

Offset	Bit Position																													
0xDE414	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x0000													
Access																			R													
Name																			HAINT													

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*		
13:0	HAINT	0x0000	R	**Channel Interrupt for channel 0 - 13.**
	When the interrupt bit for a channel x set, one or more of the interrupt flags in the USB_HCx_INT are set.			

38.7.39 USB_HAINTMSK - Host All Channels Interrupt Mask Register

The Host All Channel Interrupt Mask register works with the Host All Channel Interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel. Set bits to unmask.

Offset	Bit Position																													
0xDE418	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x0000													
Access																			RW													
Name																			HAINTMSK													

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*		
13:0	HAINTMSK	0x0000	RW	**Channel Interrupt Mask for channel 0 - 13** Set bit n to unmask channel n interrupts.

38.7.40 USB_HPRT - Host Port Control and Status Register

This register is available only in Host mode. This register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for the port. Some bits in this register can trigger an interrupt to the application through the Host

Port Interrupt bit of the Core Interrupt register (USB_GINTSTS.PRTINT). On a Port Interrupt, the application must read this register and clear the bit that caused the interrupt. For the RW1 bits, the application must write a 1 to the bit to clear the interrupt.

Offset	Bit Position																													
0xDE440	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0x0		0x0				0	0x0			0	0	0	0	0	0	0	0	0
Access														R		RW				RW	R			RW	RW1	RW	RW	R	RW	RW	RW	R
Name														PRTSPD		PRTTSTCTL				PRTPWR	PRTLNSTS			PRTRST	PRTSUSP	PRTRES	PRTOVRCURRCHNG	PRTOVRCURRACT	PRTENCHNG	PRTENA	PRTCONNDET	PRTCONNSTS

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*		
18:17	PRTSPD	0x0	R	**Port Speed**

Indicates the speed of the device attached to this port.

Value	Mode	Description
1	FS	Full speed.
2	LS	Low speed.

Bit	Name	Reset	Access	Description
16:13	PRTTSTCTL	0x0	RW	**Port Test Control**

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.

Value	Mode	Description
0	DISABLE	Test mode disabled.
1	J	Test_J mode.
2	K	Test_K mode.
3	SE0NAK	Test_SE0_NAK mode.
4	PACKET	Test_Packet mode.
5	FORCE	Test_Force_Enable.

Bit	Name	Reset	Access	Description
12	PRTPWR	0	RW	**Port Power**

The application uses this field to control power to this port. The core can clear this bit on an over current condition.

Value	Mode	Description
0	OFF	Power off.
1	ON	Power on.

Bit	Name	Reset	Access	Description
11:10	PRTLNSTS	0x0	R	**Port Line Status**
	Indicates the current logic level USB data lines Bit [10]: Logic level of D+ Bit [11]: Logic level of D-			
9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	PRTRST	0	RW	**Port Reset**
	When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete. The application must leave this bit set for at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.			
7	PRTSUSP	0	RW1	**Port Suspend**
	The application sets this bit to put this port in Suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set USB_PCGCCTL.STOPPCLK, which puts the PHY into suspend mode. The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the Port Reset bit or Port Resume bit in this register or the Resume/Remote Wakeup Detected Interrupt bit or Disconnect Detected Interrupt bit in the Core Interrupt register (USB_GINTSTS.WKUPINT or USB_GINTSTS.DISCONNINT respectively). This bit is cleared by the core even if there is no device connected to the Host.			
6	PRTRES	0	RW	**Port Resume**
	The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit. If the core detects a USB remote wakeup sequence, as indicated by the Port Resume/Remote Wakeup Detected Interrupt bit of the Core Interrupt register (USB_GINTSTS.WKUPINT), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.			
5	PRTOVR-CURRCHNG	0	RW	**Port Overcurrent Change**
	The core sets this bit when the status of the Port Overcurrent Active bit (bit 4) in this register changes.This bit can be set only by the core and the application should write 1 to clear it			
4	PRTOVRCURRACT	0	R	**Port Overcurrent Active**
	Indicates the overcurrent condition of the port. 1'b0: No overcurrent condition 1'b1: Overcurrent condition			
3	PRTENCHNG	0	RW	**Port Enable/Disable Change**
	The core sets this bit when the status of the Port Enable bit [2] of this register changes.This bit can be set only by the core and the application should write 1 to clear it.			
2	PRTENA	0	RW	**Port Enable**
	A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot Set this bit by a register write. It can only clear it to disable the port by writing 1.. This bit does not trigger any interrupt to the application. 1'b0: Port disabled 1'b1: Port enabled			
1	PRTCONNDET	0	RW	**Port Connect Detected**
	The core sets this bit when a device connection is detected to trigger an interrupt to the application using the Host Port Interrupt bit of the Core Interrupt register (GINTSTS.PRTINT).This bit can be set only by the core and the application should write 1 to clear it.The application must write a 1 to this bit to clear the interrupt.			
0	PRTCONNSTS	0	R	**Port Connect Status**
	0: No device is attached to the port. 1: A device is attached to the port.			

38.7.41 USB_HCx_CHAR - Host Channel x Characteristics Register

Offset	Bit Position																													
0xDE500	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0x00							0x0		0x0		0		0	0x0				0x000										
Access	RW1	RW1	RW	RW							RW		RW		RW		RW	RW				RW										
Name	CHENA	CHDIS	ODDFRM	DEVADDR							MC		EPTYPE		LSPDDEV		EPDIR	EPNUM				MPS										

Bit	Name	Reset	Access	Description
31	CHENA	0	RW1	**Channel Enable**
	This field is set by the application and cleared by the core. The state of this bit reflects the channel enable status.			
30	CHDIS	0	RW1	**Channel Disable**
	The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer For that channel is complete. The application must wait For the Channel Disabled interrupt before treating the channel as disabled.			
29	ODDFRM	0	RW	**Odd Frame**
	This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions. 1'b0: Even frame 1'b1: Odd frame			
28:22	DEVADDR	0x00	RW	**Device Address**
	This field selects the specific device serving as the data source or sink.			
21:20	MC	0x0	RW	**Multi Count (MC) / Error Count**
	For periodic transfers this field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration.			
19:18	EPTYPE	0x0	RW	**Endpoint Type**
	Indicates the transfer type selected.			

Value	Mode	Description
0	CONTROL	Control endpoint.
1	ISO	Isochronous endpoint.
2	BULK	Bulk endpoint.
3	INT	Interrupt endpoint.

Bit	Name	Reset	Access	Description
17	LSPDDEV	0	RW	**Low-Speed Device**
	This field is Set by the application to indicate that this channel is communicating to a low-speed device.			
16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	EPDIR	0	RW	**Endpoint Direction**
	Indicates whether the transaction is IN or OUT.			

Value	Mode	Description

Bit	Name	Reset	Access	Description
	0	OUT		Direction is OUT.
	1	IN		Direction is IN.
14:11	EPNUM	0x0	RW	**Endpoint Number**
	Indicates the endpoint number on the device serving as the data source or sink.			
10:0	MPS	0x000	RW	**Maximum Packet Size**
	Indicates the maximum packet size of the associated endpoint.			

38.7.42 USB_HCx_SPLT - Host Channel x Split Control Register

Offset	Bit Position																														
0xDE504	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0															0	0x0		0x00							0x00						
Access	RW															RW	RW		RW							RW						
Name	SPLTENA															COMPSPLT	XACTPOS		HUBADDR							PRTADDR						

Bit	Name	Reset	Access	Description
31	SPLTENA	0	RW	**Split Enable**
	The application sets this field to indicate that this channel is enabled to perform split transactions.			
30:17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	COMPSPLT	0	RW	**Do Complete Split**
	The application sets this field to request the OTG host to perform a complete split transaction.			
15:14	XACTPOS	0x0	RW	**Transaction Position**
	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. 2'b11: All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes). 2'b10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes). 2'b00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes). 2'b01: End. This is the last payload of this transaction (which is larger than 188 bytes).			
13:7	HUBADDR	0x00	RW	**Hub Address**
	This field holds the device address of the transaction translator's hub.			
6:0	PRTADDR	0x00	RW	**Port Address**
	This field is the port number of the recipient transaction translator.			

38.7.43 USB_HCx_INT - Host Channel x Interrupt Register

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (USB_GINTSTS.HCHINT) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (USB_HAINT) register to get the exact channel number for the Host Channel x

Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the USB_HAINT and USB_GINTSTS registers.

Offset	Bit Position																													
0xDE508	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0	0	0	0		0	0	0	0	0	0
Access																						RW1	RW1	RW1	RW1		RW1	RW1	RW1	RW1	RW1	RW1
Name																						DATATGLERR	FRMOVRUN	BBLERR	XACTERR		ACK	NAK	STALL	AHBERR	CHHLTD	XFERCOMPL

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	DATATGLERR	0	RW1	**Data Toggle Error**
	This bit can be set only by the core and the application should write 1 to clear it.			
9	FRMOVRUN	0	RW1	**Frame Overrun**
	This bit can be set only by the core and the application should write 1 to clear it.			
8	BBLERR	0	RW1	**Babble Error**
	This bit can be set only by the core and the application should write 1 to clear it.			
7	XACTERR	0	RW1	**Transaction Error**
	Indicates one of the following errors occurred on the USB: CRC check failure, Timeout, Bit stuff error or False EOP. This bit can be set only by the core and the application should write 1 to clear it.			
6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	ACK	0	RW1	**ACK Response Received/Transmitted Interrupt**
	This bit can be set only by the core and the application should write 1 to clear it.			
4	NAK	0	RW1	**NAK Response Received Interrupt**
	This bit can be set only by the core and the application should write 1 to clear it.			
3	STALL	0	RW1	**STALL Response Received Interrupt**
	This bit can be set only by the core and the application should write 1 to clear it.			
2	AHBERR	0	RW1	**AHB Error**
	This is generated only in DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address.			
1	CHHLTD	0	RW1	**Channel Halted**
	In DMA mode this bit indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer.			
0	XFERCOMPL	0	RW1	**Transfer Completed**
	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it.			

38.7.44 USB_HCx_INTMSK - Host Channel x Interrupt Mask Register

Offset	Bit Position																												
0xDE50C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0	0	0	0		0	0	0	0	0	0
Access																						RW	RW	RW	RW		RW	RW	RW	RW	RW	RW
Name																						DATATGLERRMSK	FRMOVRUNMSK	BBLERRMSK	XACTERRMSK		ACKMSK	NAKMSK	STALLMSK	AHBERRMSK	CHHLTDMSK	XFERCOMPLMSK

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	DATATGLERRMSK	0	RW	**Data Toggle Error Mask**
	Set to unmask DATATGLERR interrupt.			
9	FRMOVRUNMSK	0	RW	**Frame Overrun Mask**
	Set to unmask FRMOVRUN interrupt.			
8	BBLERRMSK	0	RW	**Babble Error Mask**
	Set to unmask BBLERR interrupt.			
7	XACTERRMSK	0	RW	**Transaction Error Mask**
	Set to unmask XACTERR interrupt.			
6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	ACKMSK	0	RW	**ACK Response Received/Transmitted Interrupt Mask**
	Set to unmask ACK interrupt.			
4	NAKMSK	0	RW	**NAK Response Received Interrupt Mask**
	Set to unmask NAK interrupt.			
3	STALLMSK	0	RW	**STALL Response Received Interrupt Mask**
	Set to unmask STALL interrupt.			
2	AHBERRMSK	0	RW	**AHB Error Mask**
	Set to unmask AHBERR interrupt.			
1	CHHLTDMSK	0	RW	**Channel Halted Mask**
	Set to unmask CHHLTD interrupt.			
0	XFERCOMPLMSK	0	RW	**Transfer Completed Mask**
	Set to unmask XFERCOMPL interrupt.			

38.7.45 USB_HCx_TSIZ - Host Channel x Transfer Size Register

Offset	Bit Position																															
0xDE510	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0		0x000										0x00000																		
Access		RW		RW										RW																		
Name		PID		PKTCNT										XFERSIZE																		

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:29	PID	0x0	RW	**The Application Programs This Field With the Type of**

the initial transaction. The host maintains this field For the rest of the transfer.

Value	Mode	Description
0	DATA0	DATA0 PID.
1	DATA2	DATA2 PID.
2	DATA1	DATA1 PID.
3	MDATA	MDATA (non-control) / SETUP (control) PID.

28:19	PKTCNT	0x000	RW	**Packet Count**

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as Width of Packet Counters

18:0	XFERSIZE	0x00000	RW	**Transfer Size**

For an OUT, this field is the number of data bytes the host sends during the transfer. For an IN, this field is the buffer size that the application has Reserved For the transfer. The application is expected to program this field as an integer multiple of the maximum packet size For IN transactions (periodic and non-periodic). The width of this counter is specified as Width of Transfer Size Counters

38.7.46 USB_HCx_DMAADDR - Host Channel x DMA Address Register

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions.

Offset	Bit Position																													
0xDE514	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	RW																															
Name	DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DMAADDR	0x00000000	RW	**DMA Address**
	This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction. The data for this register field is stored in RAM. Thus, the reset value is undefined (X).			

38.7.47 USB_DCFG - Device Configuration Register

This register configures the core in Device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

Offset	Bit Position																													
0xDE800	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x02																0	0	0	0x0		0x00							0	0	0x0	
Access	RW																RW	RW	RW	RW		RW							RW	RW	RW	
Name	RESVALID																ERRATICINTMSK	XCVRDLY	ENDEVOUTNAK	PERFRINT		DEVADDR							ENA32KHZSUSP	NZSTSOUTHSHK	DEVSPD	

Bit	Name	Reset	Access	Description
31:26	RESVALID	0x02	RW	**Resume Validation Period**
	This field is effective only when DCFG.ENA32KHZSusp is set. It will control the resume period when the core resumes from suspend. The core counts for RESVALID number of clock cycles to detect a valid resume when this is set			
25:16	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
15	ERRATICINTMSK	0	RW	
	Erratic Error Interrupt Mask 1'b1: Mask early suspend interrupt on erratic error 1'b0: Early suspend interrupt is generated on erratic error			
14	XCVRDLY	0	RW	
	1'b1: Enable delay between xcvr_sel and txvalid during Device chirp 1'b0: No delay between xcvr_sel and txvalid during Device chirp			
13	ENDEVOUTNAK	0	RW	**Enable Device OUT NAK**
	This bit enables setting NAK for Bulk OUT endpoints after the transfer is completed for Device mode Descriptor DMA 1'b0 : The core does not set NAK after Bulk OUT transfer complete 1'b1 : The core sets NAK after Bulk OUT transfer complete It is one time programmable after reset like any other DCFG register bits.			
12:11	PERFRINT	0x0	RW	**Periodic Frame Interval**
	Indicates the time within a frame at which the application must be notified using the End Of Periodic Frame Interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.			

Value	Mode	Description
0	80PCNT	80% of the frame interval.
1	85PCNT	85% of the frame interval.
2	90PCNT	90% of the frame interval.
3	95PCNT	95% of the frame interval.

10:4	DEVADDR	0x00	RW	**Device Address**
	The application must program this field after every SetAddress control command.			

Bit	Name	Reset	Access	Description
3	ENA32KHZSUSP	0	RW	**Enable 32 kHz Suspend Mode**
	This bit can be set only if FS PHY interface is selected. Else, this bit needs to be set to zero. When FS PHY interface is chosen and this bit is set, the core expects that the USB Core Clock during Suspend is switched from 48 MHz to 32 KHz.			
2	NZSTSOUTHSHK	0	RW	**Non-Zero-Length Status OUT Handshake**
	The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's Status stage. 1'b1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application. 1'b0: Send the received OUT packet to the application (zerolength or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the Device Endpoint Control register.			
1:0	DEVSPD	0x0	RW	**Device Speed**
	Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.			

Value	Mode	Description
2	LS	Low speed (PHY clock is 6 MHz). If you select 6 MHz LS mode, you must do a soft reset.
3	FS	Full speed (PHY clock is 48 MHz).

38.7.48 USB_DCTL - Device Control Register

Offset	Bit Position																													
0xDE804	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0	0				0	0	0	0	0		0x0		0	0	1	0
Access																RW	RW				RW	W1	W1	W1	W1		RW		R	R	RW	RW
Name																NAKONBBLE	IGNRFRMNUM				PWRONPRGDONE	CGOUTNAK	SGOUTNAK	CGNPINNAK	SGNPINNAK		TSTCTL		GOUTNAKSTS	GNPINNAKSTS	SFTDISCON	RMTWKUPSIG

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	NAKONBBLE	0	RW	**NAK on Babble Error**
	Set NAK automatically on babble (NAKONBBLE). The core sets NAK automatically for the endpoint on which babble is received.			
15	IGNRFRMNUM	0	RW	**Ignore Frame number For Isochronous End points**
	Do NOT program IGNRFRMNUM bit to 1 when the core is operating in threshold mode. When set to 0 the core transmits the packets only in the frame number in which they are intended to be transmitted. When set to 1 the core ignores the frame number, sending packets immediately as the packets are ready.			
14:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	PWRONPRGDONE	0	RW	**Power-On Programming Done**
	The application uses this bit to indicate that register programming is completed after a wake-up from Power Down mode.			
10	CGOUTNAK	0	W1	**Clear Global OUT NAK**
	A write to this field clears the Global OUT NAK.			
9	SGOUTNAK	0	W1	**Set Global OUT NAK**
	A write to this field sets the Global OUT NAK. The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set the this bit only after making sure that the Global OUT NAK Effective bit in the Core Interrupt Register (GINTSTS.GOUTNAKEFF) is cleared.			
8	CGNPINNAK	0	W1	**Clear Global Non-periodic IN NAK**
	A write to this field clears the Global Non-periodic IN NAK.			
7	SGNPINNAK	0	W1	**Set Global Non-periodic IN NAK**
	A write to this field sets the Global Non-periodic IN NAK.The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The core can also Set this bit when a timeout condition is detected on a non-periodic endpoint in shared FIFO operation. The application must Set this bit only after making sure that the Global IN NAK Effective bit in the Core Interrupt Register (GINTSTS.GINNAKEFF) is cleared			
6:4	TSTCTL	0x0	RW	**Test Control**
	Set to a non-zero value to enable test control.			

Value	Mode	Description
0	DISABLE	Test mode disabled.

Bit	Name	Reset	Access	Description
	1	J		Test_J mode.
	2	K		Test_K mode.
	3	SE0NAK		Test_SE0_NAK mode.
	4	PACKET		Test_Packet mode.
	5	FORCE		Test_Force_Enable.
3	GOUTNAKSTS	0	R	**Global OUT NAK Status**
	1'b0: A handshake is sent based on the FIFO Status and the NAK and STALL bit settings. 1'b1: No data is written to the RxFIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.			
2	GNPINNAKSTS	0	R	**Global Non-periodic IN NAK Status**
	1'b0: A handshake is sent out based on the data availability in the transmit FIFO. 1'b1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.			
1	SFTDISCON	1	RW	**Soft Disconnect**
	The application uses this bit to signal the core to do a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit. When suspended, the minimum duration for which the core must keep this bit set is 1 ms + 2.5 us. When IDLE or performing transactions, the minimum duration for which the core must keep this bit set is 2.5 us.			
0	RMTWKUPSIG	0	RW	**Remote Wakeup Signaling**
	When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the Suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1-15 ms after setting it.			

38.7.49 USB_DSTS - Device Status Register

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from Device All Interrupts (USB_DAINT) register.

Offset	Bit Position																													
0xDE808	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset									0x0		0x0000																		0	0x1		0
Access									R		R																		R	R		R
Name									DEVLNSTS		SOFFN																		ERRTICERR	ENUMSPD		SUSPSTS

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	DEVLNSTS	0x0	R	**Device Line Status** Indicates the current logic level USB data lines DevLnSts[1]: Logic level of D+ DevLnSts[0]: Logic level of D-
21:8	SOFFN	0x0000	R	**Frame Number of the Received SOF** This field contains a Frame number. This field may return a non zero value if read immediately after power on reset. In case the register bits reads non zero immediately after power on reset it does not indicate that SOF has been received from the host. The read value of this interrupt is valid only after a valid connection between host and device is established.
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	ERRTICERR	0	R	**Erratic Error** The core sets this bit to report any erratic errors (phy_rxvalid_i/phy_rxvldh_i or phy_rxactive_i is asserted For at least 2 ms, due to PHY error) seen on the UTMI+ . Because of erratic errors, DWC_otg goes into Suspended state and an interrupt is generated to the application with Early Suspend bit of the Core Interrupt register (GINTSTS.ERLYSUSP). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.
2:1	ENUMSPD	0x1	R	**Enumerated Speed** Indicates the speed at which the core has come up after speed detection through a chirp sequence.

Value	Mode	Description
2	LS	Low speed (PHY clock is running at 6 MHz).
3	FS	Full speed (PHY clock is running at 48 MHz).

0	SUSPSTS	0	R	**Suspend Status** In Device mode, this bit is Set as long as a Suspend condition is detected on the USB. The core enters the Suspended state when there is no activity on the phy_line_state_i signal For an extended period of time. The core comes out of the suspend: When there is any activity on the phy_line_state_i signal When the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RMTWKUPSIG).

38.7.50 USB_DIEPMSK - Device IN Endpoint Common Interrupt Mask Register

Offset	Bit Position																													
0xDE810	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0					0		0	0	0	0	0	0	0
Access																			RW					RW		RW	RW	RW	RW	RW	RW	RW
Name																			NAKMSK					TXFIFOUNDRNMSK		INEPNAKEFFMSK	INTKNEPMISMSK	INTKNTXFEMPMSK	TIMEOUTMSK	AHBERRMSK	EPDISBLDMSK	XFERCOMPLMSK

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	NAKMSK	0	RW	**NAK interrupt Mask**
	Set to 1 to unmask NAK Interrupt.			
12:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	TXFIFOUNDRNMSK	0	RW	**Fifo Underrun Mask**
	Set to 1 to unmask TXFIFOUNDRN Interrupt.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	INEPNAKEFFMSK	0	RW	**IN Endpoint NAK Effective Mask**
	Set to 1 to unmask INEPNAKEFF Interrupt.			
5	INTKNEPMISMSK	0	RW	**IN Token received with EP Mismatch Mask**
	Set to 1 to unmask INTKNEPMIS Interrupt.			
4	INTKNTXFEMPMSK	0	RW	**IN Token Received When TxFIFO Empty Mask**
	Set to 1 to unmask INTKNTXFEMP Interrupt.			
3	TIMEOUTMSK	0	RW	**Timeout Condition Mask**
	Set to 1 to unmask Interrupt TIMEOUT. Applies to Non-isochronous endpoints.			
2	AHBERRMSK	0	RW	**AHB Error Mask**
	Set to 1 to unmask AHBERR Interrupt.			
1	EPDISBLDMSK	0	RW	**Endpoint Disabled Interrupt Mask**
	Set to 1 to unmask EPDISBLD Interrupt.			
0	XFERCOMPLMSK	0	RW	**Transfer Completed Interrupt Mask**
	Set to 1 to unmask XFERCOMPL Interrupt.			

38.7.51 USB_DOEPMSK - Device OUT Endpoint Common Interrupt Mask Register

This register works with each of the Device OUT Endpoint Interrupt (USB_DOEP0INT/USB_DOEPx_INT) registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the USB_DOEP0INT/USB_DOEPx_INT register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Offset	Bit Position																													
0xDE814	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0	0				0		0	0	0	0	0	0	0
Access																			RW	RW				RW		RW	RW	RW	RW	RW	RW	RW
Name																			NAKMSK	BBLEERRMSK				OUTPKTERRMSK		BACK2BACKSETUP	STSPHSERCVDMSK	OUTTKNEPDISMSK	SETUPMSK	AHBERRMSK	EPDISBLDMSK	XFERCOMPLMSK

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	NAKMSK	0	RW	**NAK interrupt Mask**
	Set to 1 to unmask NAK Interrupt.			
12	BBLEERRMSK	0	RW	**Babble Error interrupt Mask**
	Set to 1 to unmask BBLEERR Interrupt.			
11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	OUTPKTERRMSK	0	RW	**OUT Packet Error Mask**
	Set to 1 to unmask OUTPKTERR Interrupt.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	BACK2BACKSETUP	0	RW	**Back-to-Back SETUP Packets Received Mask**
	Set to 1 to unmask BACK2BACKSETUP Interrupt. Applies to control OUT endpoints only.			
5	STSPHSERCVDMSK	0	RW	**Status Phase Received Mask**
	Set to 1 to unmask STSPHSERCVD Interrupt. Applies to control OUT endpoints only.			
4	OUTTKNEPDISMSK	0	RW	**OUT Token Received when Endpoint Disabled Mask**
	Set to 1 to unmask OUTTKNEPDIS Interrupt. Applies to control OUT endpoints only.			
3	SETUPMSK	0	RW	**SETUP Phase Done Mask**
	Set to 1 to unmask SETUP Interrupt. Applies to control endpoints only.			
2	AHBERRMSK	0	RW	**AHB Error**
	Set to 1 to unmask AHBERR Interrupt.			
1	EPDISBLDMSK	0	RW	**Endpoint Disabled Interrupt Mask**
	Set to 1 to unmask EPDISBLD Interrupt.			

Bit	Name	Reset	Access	Description
0	XFERCOMPLMSK	0	RW	**Transfer Completed Interrupt Mask**
	Set to 1 to unmask XFERCOMPL Interrupt.			

38.7.52 USB_DAINT - Device All Endpoints Interrupt Register

When a significant event occurs on an endpoint, a Device All Endpoints Interrupt register interrupts the application using the Device OUT Endpoints Interrupt bit or Device IN Endpoints Interrupt bit of the Core Interrupt register (USB_GINTSTS.OEPINT or USB_GINTSTS.IEPINT, respectively). There is one interrupt bit per endpoint. For a bidirectional endpoint, the corresponding IN and

OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint Interrupt register (USB_DIEP0INT/USB_DIEPx_INT, USB_DOEP0INT/USB_DOEPx_INT).

Offset	Bit Position																														
0xDE818	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset										0	0	0	0	0	0	0										0	0	0	0	0	0	0
Access										R	R	R	R	R	R	R										R	R	R	R	R	R	R
Name										OUTEPINT6	OUTEPINT5	OUTEPINT4	OUTEPINT3	OUTEPINT2	OUTEPINT1	OUTEPINT0										INEPINT6	INEPINT5	INEPINT4	INEPINT3	INEPINT2	INEPINT1	INEPINT0

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	OUTEPINT6	0	R	**OUT Endpoint 6 Interrupt Bit**
	This bit is set when on or more of the interrupt flags in USB_DOEP6_INT are set.			
21	OUTEPINT5	0	R	**OUT Endpoint 5 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DOEP5_INT are set.			
20	OUTEPINT4	0	R	**OUT Endpoint 4 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DOEP4_INT are set.			
19	OUTEPINT3	0	R	**OUT Endpoint 3 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DOEP3_INT are set.			
18	OUTEPINT2	0	R	**OUT Endpoint 2 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DOEP2_INT are set.			
17	OUTEPINT1	0	R	**OUT Endpoint 1 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DOEP1_INT are set.			
16	OUTEPINT0	0	R	**OUT Endpoint 0 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DOEP0INT are set.			
15:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	INEPINT6	0	R	**IN Endpoint 6 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP6_INT are set.			
5	INEPINT5	0	R	**IN Endpoint 5 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP5_INT are set.			
4	INEPINT4	0	R	**IN Endpoint 4 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP4_INT are set.			
3	INEPINT3	0	R	**IN Endpoint 3 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP3_INT are set.			
2	INEPINT2	0	R	**IN Endpoint 2 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP2_INT are set.			

Bit	Name	Reset	Access	Description
1	INEPINT1	0	R	**IN Endpoint 1 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP1_INT are set.			
0	INEPINT0	0	R	**IN Endpoint 0 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP0INT are set.			

38.7.53 USB_DAINTMSK - Device All Endpoints Interrupt Mask Register

The Device Endpoint Interrupt Mask register works with the Device Endpoint Interrupt register to interrupt the application when an event occurs on a device endpoint. However, the Device All Endpoints Interrupt (USB_DAINT) register bit corresponding to that interrupt is still set.

Offset	Bit Position																														
0xDE81C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset										0	0	0	0	0	0	0										0	0	0	0	0	0	0
Access										RW	RW	RW	RW	RW	RW	RW										RW	RW	RW	RW	RW	RW	RW
Name										OUTEPMSK6	OUTEPMSK5	OUTEPMSK4	OUTEPMSK3	OUTEPMSK2	OUTEPMSK1	OUTEPMSK0										INEPMSK6	INEPMSK5	INEPMSK4	INEPMSK3	INEPMSK2	INEPMSK1	INEPMSK0

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	OUTEPMSK6	0	RW	**OUT Endpoint 6 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.OUTEPINT6.			
21	OUTEPMSK5	0	RW	**OUT Endpoint 5 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.OUTEPINT5.			
20	OUTEPMSK4	0	RW	**OUT Endpoint 4 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.OUTEPINT4.			
19	OUTEPMSK3	0	RW	**OUT Endpoint 3 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.OUTEPINT3.			
18	OUTEPMSK2	0	RW	**OUT Endpoint 2 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.OUTEPINT2.			
17	OUTEPMSK1	0	RW	**OUT Endpoint 1 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.OUTEPINT1.			
16	OUTEPMSK0	0	RW	**OUT Endpoint 0 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.OUTEPINT0.			
15:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	INEPMSK6	0	RW	**IN Endpoint 6 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.INEPINT6.			
5	INEPMSK5	0	RW	**IN Endpoint 5 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.INEPINT5.			
4	INEPMSK4	0	RW	**IN Endpoint 4 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.INEPINT4.			
3	INEPMSK3	0	RW	**IN Endpoint 3 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.INEPINT3.			

Bit	Name	Reset	Access	Description
2	INEPMSK2	0	RW	**IN Endpoint 2 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.INEPINT2.			
1	INEPMSK1	0	RW	**IN Endpoint 1 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.INEPINT1.			
0	INEPMSK0	0	RW	**IN Endpoint 0 Interrupt mask Bit**
	Set to 1 to unmask USB_DAINT.INEPINT0.			

38.7.54 USB_DVBUSDIS - Device VBUS Discharge Time Register

Offset	Bit Position																													
0xDE828	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x17D7						
Access																									RW						
Name																									DVBUSDIS						

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	DVBUSDIS	0x17D7	RW	**Device VBUS Discharge Time**
	Specifies the VBUS discharge time after VBUS pulsing during SRP. This value equals VBUS discharge time in PHY clocks / 1024. Depending on your VBUS load, this value can need adjustment.			

38.7.55 USB_DVBUSPULSE - Device VBUS Pulsing Time Register

Offset	Bit Position																													
0xDE82C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x5B8				
Access																											RW				
Name																											DVBUSPULSE				

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	DVBUSPULSE	0x5B8	RW	**Device VBUS Pulsing Time** Specifies the VBUS pulsing time during SRP. This value equals: VBUS pulsing time in USB Core Clocks / 1, 024

38.7.56 USB_DTHRCTL - Device Threshold Control Register

Offset	Bit Position																													
0xDE830	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					1		0x008									0				0x0		0x008								0	0
Access					RW		RW									RW				RW		RW								RW	RW
Name					ARBPRKEN		RXTHRLEN									RXTHREN				AHBTHRRATIO		TXTHRLEN								ISOTHREN	NONISOTHREN

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	ARBPRKEN	1	RW	**Arbiter Parking Enable**
	This bit controls internal DMA arbiter parking For IN endpoints. When thresholding is enabled and this bit is Set to one, Then the arbiter parks on the IN endpoint For which there is a token received on the USB. This is done to avoid getting into underrun conditions. By Default the parking is enabled.			
26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25:17	RXTHRLEN	0x008	RW	**Receive Threshold Length**
	This field specifies Receive thresholding size in DWORDS. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight DWORDS. The recommended value For ThrLen is to be the same as the programmed AHB Burst Length (GAHBCFG.HBSTLEN).			
16	RXTHREN	0	RW	**Receive Threshold Enable**
	When this bit is set, the core enables thresholding in the receive direction. Note: We recommends that you do not enable RXTHREN, because it may cause issues in the RxFIFO especially during error conditions such as RxError and Babble.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:11	AHBTHRRATIO	0x0	RW	**AHB Threshold Ratio**
	These bits define the ratio between the AHB threshold and the MAC threshold for the transmit path only. The AHB threshold always remains less than or equal to the USB threshold, because this does not increase overhead. Both the AHB and the MAC threshold must be DWORD-aligned. The application needs to program TXTHRLEN and the AHBTHRRATIO to make the AHB Threshold value DWORD aligned. If the AHB threshold value is not DWORD aligned, the core might not behave correctly. When programming the TXTHRLEN and AHBTHRRATIO, the application must ensure that the minimum AHB threshold value does not go below 8 DWORDS to meet the USB turnaround time requirements.			

Value	Mode	Description
0	DIV1	AHB threshold = MAC threshold.
1	DIV2	AHB threshold = MAC threshold / 2.
2	DIV4	AHB threshold = MAC threshold / 4.
3	DIV8	AHB threshold = MAC threshold / 8.

Bit	Name	Reset	Access	Description
10:2	TXTHRLEN	0x008	RW	**Transmit Threshold Length** This field specifies Transmit thresholding size in DWORDS. This also forms the MAC threshold and specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmit on the USB. The threshold length has to be at least eight DWORDS when the value of AHBTHRRATIO is 2'h00. In case the AHBTHRRATIO is non zero the application needs to ensure that the AHB Threshold value does not go below the recommended eight DWORD. This field controls both isochronous and non-isochronous IN endpoint thresholds. The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (GAHBCFG.HBSTLEN).
1	ISOTHREN	0	RW	**ISO IN Endpoints Threshold Enable** When this bit is Set, the core enables thresholding For isochronous IN endpoints.
0	NONISOTHREN	0	RW	**Non-ISO IN Endpoints Threshold Enable** When this bit is Set, the core enables thresholding For Non Isochronous IN endpoints.

38.7.57 USB_DIEPEMPMSK - Device IN Endpoint FIFO Empty Interrupt Mask Register

Offset	Bit Position																													
0xDE834	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x0000							
Access																									RW							
Name																									INEPTXFEMPMSK							

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	INEPTXFEMPMSK	0x0000	RW	**IN EP Tx FIFO Empty Interrupt Mask Bits** These bits acts as mask bits for USB_DIEP0INT.TXFEMP/USB_DIEPx_INT.TXFEMP interrupt. One bit per IN Endpoint: Bit 0 for IN EP 0, bit 6 for IN EP 6.

38.7.58 USB_DIEP0CTL - Device Control IN Endpoint 0 Control Register

Offset	Bit Position																													
0xDE900	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0			0	0	0x0				0		0x0		0		1														0x0	
Access	RW1	RW1			W1	W1	RW				RW1		R		R		R														RW	
Name	EPENA	EPDIS			SNAK	CNAK	TXFNUM				STALL		EPTYPE		NAKSTS		USBACTEP														MPS	

Bit	Name	Reset	Access	Description
31	EPENA	0	RW1	**Endpoint Enable**
	In DMA mode this bit indicates that data is ready to be transmitted on the endpoint. The core clears this bit before setting the following interrupts on this endpoint: Endpoint Disabled, Transfer Completed.			
30	EPDIS	0	RW1	**Endpoint Disable**
	The application sets this bit to stop transmitting data on an endpoint, even before the transfer For that endpoint is complete. The application must wait For the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled Interrupt. The application must Set this bit only If Endpoint Enable is already Set For this endpoint.			
29:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	SNAK	0	W1	**Set NAK**
	A write to this bit sets the NAK bit For the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit For an endpoint after a SETUP packet is received on that endpoint.			
26	CNAK	0	W1	**Clear NAK**
	A write to this bit clears the NAK bit For the endpoint.			
25:22	TXFNUM	0x0	RW	**TxFIFO Number**
	For Shared FIFO operation, this value is always Set to 0, indicating that control IN endpoint 0 data is always written in the Non-Periodic Transmit FIFO. For Dedicated FIFO operation, this value is Set to the FIFO number that is assigned to IN Endpoint 0.			
21	STALL	0	RW1	**Handshake**
	The application can only Set this bit, and the core clears it, when a SETUP token is received For this endpoint. If a NAK bit, Global Nonperiodic IN NAK, or Global OUT NAK is Set along with this bit, the STALL bit takes priority.			
20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:18	EPTYPE	0x0	R	**Endpoint Type**
	Hardcoded to 00 for control.			
17	NAKSTS	0	R	**NAK Status**
	Indicates the following: 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status 1'b1: The core is transmitting NAK handshakes on this endpoint. When this bit is Set, either by the application or core, the core stops transmitting data, even If there is data available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.			
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	USBACTEP	1	R	**USB Active Endpoint** This bit is always SET to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.
14:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	MPS	0x0	RW	**Maximum Packet Size** Applies to IN and OUT endpoints. The application must program this field with the maximum packet size For the current logical endpoint.

Value	Mode	Description
0	64B	64 bytes.
1	32B	32 bytes.
2	16B	16 bytes.
3	8B	8 bytes.

38.7.59 USB_DIEP0INT - Device IN Endpoint 0 Interrupt Register

This register indicates the status of endpoint 0 with respect to USB- and AHB-related events. The application must read this register when the IN Endpoints Interrupt bit of the Core Interrupt register (USB_GINTSTS.IEPINT) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (USB_DAINT) register to get the exact endpoint number for the Device

Endpoint Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the USB_DAINT and USB_GINTSTS registers.

Offset	Bit Position																													
0xDE908	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0	0	0			0	1	0	0	0	0	0	0	0
Access																			RW1	RW1	RW1			RW1	R	RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																			NAKINTRPT	BBLEERR	PKTDRPSTS			TXFIFOUNDRN	TXFEMP	INEPNAKEFF	INTKNEPMIS	INTKNTXFEMP	TIMEOUT	AHBERR	EPDISBLD	XFERCOMPL

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	NAKINTRPT	0	RW1	**NAK Interrupt**
	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.			
12	BBLEERR	0	RW1	**Babble Interrupt**
	The core generates this interrupt when babble is received for the endpoint.			
11	PKTDRPSTS	0	RW1	**Packet Drop Status**
	This bit indicates to the application that an ISO OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.			
10:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	TXFIFOUNDRN	0	RW1	**Fifo Underrun**
	This bit is valid only if thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint.			
7	TXFEMP	1	R	**Transmit FIFO Empty**
	This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (USB_GAHBCFG.NPTXFEMPLVL).			
6	INEPNAKEFF	0	RW1	**IN Endpoint NAK Effective**
	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to USB_DIEP0CTL.CNAK. This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.			
5	INTKNEPMIS	0	RW1	**IN Token Received with EP Mismatch**
	Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.			
4	INTKNTXFEMP	0	RW1	**IN Token Received When TxFIFO is Empty**
	Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.			

Bit	Name	Reset	Access	Description
3	TIMEOUT	0	RW1	**Timeout Condition** Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.
2	AHBERR	0	RW1	**AHB Error** This is generated in DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.
1	EPDISBLD	0	RW1	**Endpoint Disabled Interrupt** This bit indicates that the endpoint is disabled per the application's request.
0	XFERCOMPL	0	RW1	**Transfer Completed Interrupt** This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

38.7.60 USB_DIEP0TSIZ - Device IN Endpoint 0 Transfer Size Register

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using Endpoint Enable bit of the Device Control Endpoint 0 Control register (USB_DIEP0CTL.EPENA), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit. Nonzero endpoints use the registers for endpoints 1-6.

Offset	Bit Position																														
0xDE910	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset												0x0																	0x00		
Access												RW																	RW		
Name												PKTCNT																	XFERSIZE		

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*		
20:19	PKTCNT	0x0	RW	**Packet Count** Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.
18:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	XFERSIZE	0x00	RW	**Transfer Size** Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.

38.7.61 USB_DIEP0DMAADDR - Device IN Endpoint 0 DMA Address Register

Offset	Bit Position																													
0xDE914	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	RW																															
Name	DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DMAADDR	0x00000000	RW	
	Holds the start address of the external memory for fetching endpoint data. For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. The data for this register field is stored in RAM. Thus, the reset value is undefined (X).			

38.7.62 USB_DIEP0TXFSTS - Device IN Endpoint Transmit FIFO Status Register 0

Offset	Bit Position																													
0xDE918	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0200															
Access																	R															
Name																	SPCAVAIL															

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	SPCAVAIL	0x0200	R	**IN Endpoint TxFIFO Space Avail**
	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words.			

38.7.63 USB_DIEPx_CTL - Device Control IN Endpoint x+1 Control Register

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Offset	Bit Position																													
0xDE920	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0x0				0		0x0		0	0	0					0x000										
Access	RW1	RW1	W1	W1	W1	W1	RW				RW1		RW		R	R	RW					RW										
Name	EPENA	EPDIS	SETD1PIDOF	SETD0PIDEF	SNAK	CNAK	TXFNUM				STALL		EPTYPE		NAKSTS	DPIDEOF	USBACTEP					MPS										

Bit	Name	Reset	Access	Description
31	EPENA	0	RW1	**Endpoint Enable**
	In DMA mode for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. The core clears this bit before setting any of the following interrupts on this endpoint: SETUP Phase Done, Endpoint Disabled, Transfer Completed. For control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.			
30	EPDIS	0	RW1	**Endpoint Disable**
	The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.			
29	SETD1PIDOF	0	W1	**Set DATA1 PID / Odd Frame**
	For bulk and interrupt endpoints writing this field sets the Endpoint Data PID / Even or Odd Frame (DPIDEOF) field in this register to DATA1ODD. For isochronous endpoints writing this field sets the Endpoint Data PID / Even or Odd Frame (DPIDEOF) field to odd (DATA1ODD).			
28	SETD0PIDEF	0	W1	**Set DATA0 PID / Even Frame**
	For bulk and interrupt endpoints writing this field sets the Endpoint Data PID / Even or Odd Frame (DPIDEOF) field in this register to DATA0EVEN. For isochronous endpoints writing this field sets the Endpoint Data PID / Even or Odd Frame (DPIDEOF) field to odd (DATA0EVEN).			
27	SNAK	0	W1	**Set NAK**
	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.			
26	CNAK	0	W1	**Clear NAK**
	A write to this bit clears the NAK bit for the endpoint.			
25:22	TXFNUM	0x0	RW	**TxFIFO Number**
	These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.			
21	STALL	0	RW1	**Handshake**
	For bulk and interrupt endpoints: The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. In this case only the application can clear this bit, never the core. When control endpoint: The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.			

Bit	Name	Reset	Access	Description
20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:18	EPTYPE	0x0	RW	**Endpoint Type**

This is the transfer type supported by this logical endpoint.

Value	Mode	Description
0	CONTROL	Control Endpoint.
1	ISO	Isochronous Endpoint.
2	BULK	Bulk Endpoint.
3	INT	Interrupt Endpoint.

17	NAKSTS	0	R	**NAK Status**

When this bit is 0 the core is transmitting non-NAK handshakes based on the FIFO status. When this bit is 1 the core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit the core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. For non-isochronous IN endpoints the core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. For isochronous IN endpoints the core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

16	DPIDEOF	0	R	**Endpoint Data PID / Even or Odd Frame**

For interrupt/bulk endpoints this field contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SETD1PIDOF and SETD0PIDEF fields of this register to program either DATA0 or DATA1 PID. For isochronous endpoints, this field indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SETD0PIDEF and SETD1PIDOF fields in this register.

Value	Mode	Description
0	DATA0EVEN	DATA0 PID / Even Frame.
1	DATA1ODD	DATA1 PID / Odd Frame.

15	USBACTEP	0	RW	**USB Active Endpoint**

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

14:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:0	MPS	0x000	RW	**Maximum Packet Size**

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

38.7.64 USB_DIEPx_INT - Device IN Endpoint x+1 Interrupt Register

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the IN Endpoints Interrupt bit of the Core Interrupt register (USB_GINTSTS.IEPINT) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (USB_DAINT) register to get the exact endpoint number for the Device

Endpoint x+1 Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the USB_DAINT and USB_GINTSTS registers.

Offset	Bit Position																													
0xDE928	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0	0	0			0	1	0	0	0	0	0	0	0
Access																			RW1	RW1	RW1			RW1	R	RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																			NAKINTRPT	BBLEERR	PKTDRPSTS			TXFIFOUNDRN	TXFEMP	INEPNAKEFF	INTKNEPMIS	INTKNTXFEMP	TIMEOUT	AHBERR	EPDISBLD	XFERCOMPL

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	NAKINTRPT	0	RW1	**NAK Interrupt**
	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.			
12	BBLEERR	0	RW1	**Babble Interrupt**
	The core generates this interrupt when babble is received for the endpoint.			
11	PKTDRPSTS	0	RW1	**Packet Drop Status**
	This bit indicates to the application that an ISO OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.			
10:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	TXFIFOUNDRN	0	RW1	**Fifo Underrun**
	This bit is valid only if thresholding is enabled. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint.			
7	TXFEMP	1	R	**Transmit FIFO Empty**
	This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (USB_GAHBCFG.NPTXFEMPLVL).			
6	INEPNAKEFF	0	RW1	**IN Endpoint NAK Effective**
	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to USB_DIEPx_CTL.CNAK. This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.			
5	INTKNEPMIS	0	RW1	**IN Token Received with EP Mismatch**
	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.			
4	INTKNTXFEMP	0	RW1	**IN Token Received When TxFIFO is Empty**
	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.			

Bit	Name	Reset	Access	Description
3	TIMEOUT	0	RW1	**Timeout Condition** Applies only to Control IN endpoints. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.
2	AHBERR	0	RW1	**AHB Error** This is generated only in DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.
1	EPDISBLD	0	RW1	**Endpoint Disabled Interrupt** This bit indicates that the endpoint is disabled per the application's request.
0	XFERCOMPL	0	RW1	**Transfer Completed Interrupt** This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

38.7.65 USB_DIEPx_TSIZ - Device IN Endpoint x+1 Transfer Size Register

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint x+1 Control register (USB_DIEPx_CTL.EPENA), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Offset	Bit Position																														
0xDE930	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0		0x000										0x00000																		
Access		RW		RW										RW																		
Name		MC		PKTCNT										XFERSIZE																		

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:29	MC	0x0	RW	**Multi Count** For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.
28:19	PKTCNT	0x000	RW	**Packet Count** Indicates the total number of USB packets that constitute the Transfer Size amount of data. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.
18:0	XFERSIZE	0x00000	RW	**Transfer Size** Indicates the transfer size in bytes. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.

38.7.66 USB_DIEPx_DMAADDR - Device IN Endpoint x+1 DMA Address Register

Offset	Bit Position																													
0xDE934	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	RW																															
Name	DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DMAADDR	0x00000000	RW	
	Holds the start address of the external memory for fetching endpoint data. For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. The data for this register field is stored in RAM. Thus, the reset value is undefined (X).			

38.7.67 USB_DIEPx_DTXFSTS - Device IN Endpoint Transmit FIFO Status Register 1

Offset	Bit Position																													
0xDE938	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0200															
Access																	R															
Name																	SPCAVAIL															

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	SPCAVAIL	0x0200	R	**IN Endpoint TxFIFO Space Avail**
	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words.			

38.7.68 USB_DOEP0CTL - Device Control OUT Endpoint 0 Control Register

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Offset	Bit Position																													
0xDEB00	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0			0	0					0	0	0x0		0		1														0x0	
Access	RW1	R			W1	W1					RW1	RW	R		R		R														R	
Name	EPENA	EPDIS			SNAK	CNAK					STALL	SNP	EPTYPE		NAKSTS		USBACTEP														MPS	

Bit	Name	Reset	Access	Description
31	EPENA	0	RW1	**Endpoint Enable**
	In DMA mode this bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears this bit before setting any of the following interrupts on this endpoint: SETUP Phase Done, Endpoint Disabled, Transfer Completed. In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory.			
30	EPDIS	0	R	**Endpoint Disable**
	This bit is always 0. The application cannot disable control OUT endpoint 0.			
29:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
27	SNAK	0	W1	**Set NAK**
	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set bit on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.			
26	CNAK	0	W1	**Clear NAK**
	A write to this bit clears the NAK bit for the endpoint.			
25:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	STALL	0	RW1	**Handshake**
	The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.			
20	SNP	0	RW	**Snoop Mode**
	This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.			
19:18	EPTYPE	0x0	R	**Endpoint Type**
	Hardcoded to 0. Endpoint 0 is always a control endpoint.			
17	NAKSTS	0	R	**NAK Status**
	When this bit is 0 the core is transmitting non-NAK handshakes based on the FIFO status. When this bit is 1 the core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.			
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	USBACTEP	1	R	**USB Active Endpoint**
	This bit is always 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.			
14:2	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
1:0	MPS	0x0	R	**Maximum Packet Size**
	The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN Endpoint 0.			

Value	Mode	Description
0	64B	64 bytes.
1	32B	32 bytes.
2	16B	16 bytes.
3	8B	8 bytes.

38.7.69 USB_DOEP0INT - Device OUT Endpoint 0 Interrupt Register

This register indicates the status of endpoint 0 with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit of the Core Interrupt register (USB_GINTSTS.OEPINT) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (USB_DAINT) register to get the exact endpoint number for the Device

Endpoint Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the USB_DAINT and USB_GINTSTS registers.

Offset	Bit Position																													
0xDEB08	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0		0	0	0			0		0	0	0	0	0	0	0
Access																	RW1		RW1	RW1	RW1			RW1		RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																	STUPPKTRCVD		NAKINTRPT	BBLEERR	PKTDRPSTS			OUTPKTERR		BACK2BACKSETUP	STSPHSERCVD	OUTTKNEPDIS	SETUP	AHBERR	EPDISBLD	XFERCOMPL

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	STUPPKTRCVD	0	RW1	
	Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used. 1'b0: No Setup packet received 1'b1: Setup packet received			
14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	NAKINTRPT	0	RW1	**NAK Interrupt**
	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.			
12	BBLEERR	0	RW1	**NAK Interrupt**
	The core generates this interrupt when babble is received for the endpoint.			
11	PKTDRPSTS	0	RW1	**Packet Drop Status**
	This bit indicates to the application that an ISO OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.			
10:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	OUTPKTERR	0	RW1	**OUT Packet Error**
	This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	BACK2BACKSETUP	0	RW1	**Back-to-Back SETUP Packets Received**
	This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.			

Bit	Name	Reset	Access	Description
5	STSPHSERCVD	0	RW1	**Status Phase Received For Control Write**
	This interrupt is valid only for Control OUT endpoints. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase.			
4	OUTTKNEPDIS	0	RW1	**OUT Token Received When Endpoint Disabled**
	Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.			
3	SETUP	0	RW1	**Setup Phase Done**
	Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.			
2	AHBERR	0	RW1	**AHB Error**
	This is generated only in DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.			
1	EPDISBLD	0	RW1	**Endpoint Disabled Interrupt**
	This bit indicates that the endpoint is disabled per the application's request.			
0	XFERCOMPL	0	RW1	**Transfer Completed Interrupt**
	This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.			

38.7.70 USB_DOEP0TSIZ - Device OUT Endpoint 0 Transfer Size Register

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint x+1 Control register (USB_DOEPx_CTL.EPENA), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Offset	Bit Position																															
0xDEB10	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0											0																0x00			
Access		RW											RW																RW			
Name		SUPCNT											PKTCNT																XFERSIZE			

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:29	SUPCNT	0x0	RW	**SETUP Packet Count** This field specifies the number of back-to-back SETUP data packets the endpoint can receive. 2'b01: 1 packet 2'b10: 2 packets 2'b11: 3 packets
28:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	PKTCNT	0	RW	**Packet Count** This field is decremented to zero after a packet is written into the RxFIFO.
18:7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:0	XFERSIZE	0x00	RW	**Transfer Size** Indicates the transfer size in bytes For endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.

38.7.71 USB_DOEP0DMAADDR - Device OUT Endpoint 0 DMA Address Register

Offset	Bit Position																															
0xDEB14	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	RW																															
Name	DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DMAADDR	0x00000000	RW	

Holds the start address of the external memory for storing endpoint data. For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. The data for this register field is stored in RAM. Thus, the reset value is undefined (X).

38.7.72 USB_DOEPx_CTL - Device Control OUT Endpoint x+1 Control Register

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Offset	Bit Position																													
0xDEB20	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0					0	0	0x0		0	0	0					0x000										
Access	RW1	RW1	W1	W1	W1	W1					RW1	RW	RW		R	R	RW					RW										
Name	EPENA	EPDIS	SETD1PIDOF	SETD0PIDEF	SNAK	CNAK					STALL	SNP	EPTYPE		NAKSTS	DPIDEOF	USBACTEP					MPS										

Bit	Name	Reset	Access	Description
31	EPENA	0	RW1	**Endpoint Enable**
	In DMA mode this bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears this bit before setting any of the following interrupts on this endpoint: SETUP Phase Done, Endpoint Disabled, Transfer Completed. For control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.			
30	EPDIS	0	RW1	**Endpoint Disable**
	The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.			
29	SETD1PIDOF	0	W1	**Set DATA1 PID / Odd Frame**
	For bulk and interrupt endpoints writing this field sets the Endpoint Data PID / Even or Odd Frame (DPIDEOF) field in this register to DATA1ODD. For isochronous endpoints writing this field sets the Endpoint Data PID / Even or Odd Frame (DPIDEOF) field to odd (DATA1ODD).			
28	SETD0PIDEF	0	W1	**Set DATA0 PID / Even Frame**
	For bulk and interrupt endpoints writing this field sets the Endpoint Data PID / Even or Odd Frame (DPIDEOF) field in this register to DATA0EVEN. For isochronous endpoints writing this field sets the Endpoint Data PID / Even or Odd Frame (DPIDEOF) field to odd (DATA0EVEN).			
27	SNAK	0	W1	**Set NAK**
	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.			
26	CNAK	0	W1	**Clear NAK**
	A write to this bit clears the NAK bit for the endpoint.			
25:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21	STALL	0	RW1	**STALL Handshake**
	For non-control, non-isochronous endpoints: The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. For control endpoints: The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.			

Bit	Name	Reset	Access	Description
20	SNP	0	RW	**Snoop Mode**
	This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.			
19:18	EPTYPE	0x0	RW	**Endpoint Type**
	This is the transfer type supported by this logical endpoint.			

Value	Mode	Description
0	CONTROL	Control Endpoint.
1	ISO	Isochronous Endpoint.
2	BULK	Bulk Endpoint.
3	INT	Interrupt Endpoint.

17	NAKSTS	0	R	**NAK Status**
	When this bit is 0 the core is transmitting non-NAK handshakes based on the FIFO status. When this bit is 1 the core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit the core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.			
16	DPIDEOF	0	R	**Endpoint Data PID / Even-odd Frame**
	For interrupt/bulk endpoints: Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application use the SETD1PIDOF and SETD0PIDEF fields of this register to program either DATA0 or DATA1 PID. For isochronous endpoints: Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SETD1PIDOF and SETD0PIDEF fields in this register.			

Value	Mode	Description
0	DATA0EVEN	DATA0 PID / Even Frame.
1	DATA1ODD	DATA1 PID / Odd Frame.

15	USBACTEP	0	RW	**USB Active Endpoint**
	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.			
14:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:0	MPS	0x000	RW	**Maximum Packet Size**
	The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.			

38.7.73 USB_DOEPx_INT - Device OUT Endpoint x+1 Interrupt Register

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit of the Core Interrupt register (USB_GINTSTS.OEPINT) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (USB_DAINT) register to get the exact endpoint number for the Device

Endpoint Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the USB_DAINT and USB_GINTSTS registers.

Offset	Bit Position																													
0xDEB28	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0		0	0	0			0		0	0	0	0	0	0	0
Access																	RW		RW1	RW1	RW1			RW1		RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																	STUPPKTRCVD		NAKINTRPT	BBLEERR	PKTDRPSTS			OUTPKTERR		BACK2BACKSETUP	STSPHSERCVD	OUTTKNEPDIS	SETUP	AHBERR	EPDISBLD	XFERCOMPL

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	STUPPKTRCVD	0	RW	
	Applicable for Control OUT Endpoints in only in the Buffer DMA Mode Set by the DWC_otg core, this bit indicates that this buffer holds 8 bytes of setup data. There is only one Setup packet per buffer. On receiving a Setup packet, the DWC_otg core closes the buffer and disables the corresponding endpoint. The application has to re-enable the endpoint to receive any OUT data for the Control Transfer and reprogram the buffer start address. Note: Because of the above behavior, the DWC_otg core can receive any number of back to back setup packets and one buffer for every setup packet is used. 1'b0: No Setup packet received 1'b1: Setup packet received			
14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13	NAKINTRPT	0	RW1	**NAK Interrupt**
	The core generates this interrupt when a NAK is transmitted or received by the device.			
12	BBLEERR	0	RW1	**Babble Error**
	The core generates this interrupt when babble is received for the endpoint.			
11	PKTDRPSTS	0	RW1	**Packet Drop Status**
	This bit indicates to the application that an ISO OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.			
10:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	OUTPKTERR	0	RW1	**OUT Packet Error**
	This interrupt is valid only when thresholding is enabled. This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	BACK2BACKSETUP	0	RW1	**Back-to-Back SETUP Packets Received**
	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.			

Bit	Name	Reset	Access	Description
5	STSPHSERCVD	0	RW1	**Status Phase Received For Control Write**
	This interrupt is valid only for Control OUT endpoints . This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase.			
4	OUTTKNEPDIS	0	RW1	**OUT Token Received When Endpoint Disabled**
	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.			
3	SETUP	0	RW1	**Setup Phase Done**
	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.			
2	AHBERR	0	RW1	**AHB Error**
	This is generated only in DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.			
1	EPDISBLD	0	RW1	**Endpoint Disabled Interrupt**
	This bit indicates that the endpoint is disabled per the application's request.			
0	XFERCOMPL	0	RW1	**Transfer Completed Interrupt**
	This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.			

38.7.74 USB_DOEPx_TSIZ - Device OUT Endpoint x+1 Transfer Size Register

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint x+1 Control register (USB_DOEPx_CTL.EPENA), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Offset	Bit Position																														
0xDEB30	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0		0x000										0x00000																	
Access		R		RW										RW																	
Name		RXDPIDSUPCNT		PKTCNT										XFERSIZE																	

Bit	Name	Reset	Access	Description
31	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
30:29	RXDPIDSUPCNT	0x0	R	**Receive Data PID / SETUP Packet Count** For isochronous OUT endpoints: This is the data PID received in the last packet for this endpoint. For control OUT Endpoints: This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

Value	Mode	Description
0	DATA0	DATA0 PID.
1	DATA2	DATA2 PID / 1 Packet.
2	DATA1	DATA1 PID / 2 Packets.
3	MDATA	MDATA PID / 3 Packets.

Bit	Name	Reset	Access	Description
28:19	PKTCNT	0x000	RW	**Packet Count** This field is decremented to zero after a packet is written into the RxFIFO.
18:0	XFERSIZE	0x00000	RW	**Transfer Size** Indicates the transfer size in bytes. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.

38.7.75 USB_DOEPx_DMAADDR - Device OUT Endpoint x+1 DMA Address Register

Offset	Bit Position																															
0xDEB34	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	RW																															
Name	DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DMAADDR	0x00000000	RW	

Holds the start address of the external memory for storing endpoint data. For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. The data for this register field is stored in RAM. Thus, the reset value is undefined (X).

38.7.76 USB_PCGCCTL - Power and Clock Gating Control Register

Offset	Bit Position																														
0xDEE00	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																								0		0			0	0	0	0
Access																								R		R			RW	RW	RW	RW
Name																								RESETAFTERSUSP		PHYSLEEP			RSTPDWNMODULE	PWRCLMP	GATEHCLK	STOPPCLK

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	RESETAFTERSUSP	0	R	**Reset after suspend**
	When exiting EM2, this bit needs to be set in host mode before clamp is removed if the host needs to issue reset after suspend. If this bit is not set, then the host issues resume after suspend. This bit is not applicable in device mode and when EM2 is not used.			
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	PHYSLEEP	0	R	**PHY In Sleep**
	Indicates that the PHY is in Sleep State.			
5:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	RSTPDWNMODULE	0	RW	**Reset Power-Down Modules**
	The application sets this bit to reset the part of the USB that is powered down during EM2. The application clears this bit to release reset after an waking up from EM2 when the PHY clock is back at 48/6 MHz. Accessing core registers is possible only when this bit is set to 0.			
2	PWRCLMP	0	RW	**Power Clamp**
	The application sets this bit before the power is turned off to clamp the signals between the power-on modules and the power-off modules of the USB core. The application clears the bit to disable the clamping.			
1	GATEHCLK	0	RW	**Gate HCLK**
	The application sets this bit to gate the clock (HCLK) to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.			
0	STOPPCLK	0	RW	**Stop PHY clock**
	The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.			

38.7.77 USB_FIFO0Dx - Device EP 0/Host Channel 0 FIFO (Actionable Reads)

This register, available in both Host and Device modes, is used to read or write the FIFO space for endpoint 0 or channel 0, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xDF000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO0D																															

Bit	Name	Reset	Access	Description
31:0	FIFO0D	0xXXXXXXXX	RWH	**Device EP 0/Host Channel 0 FIFO**
	FIFO 0 push/pop region. Used in slave mode.			

38.7.78 USB_FIFO1Dx - Device EP 1/Host Channel 1 FIFO (Actionable Reads)

This register, available in both Host and Device modes, is used to read or write the FIFO space for endpoint 1 or channel 1, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE0000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO1D																															

Bit	Name	Reset	Access	Description
31:0	FIFO1D	0xXXXXXXXX	RWH	**Device EP 1/Host Channel 1 FIFO**
	FIFO 1 push/pop region. Used in slave mode.			

38.7.79 USB_FIFO2Dx - Device EP 2/Host Channel 2 FIFO (Actionable Reads)

This register, available in both Host and Device modes, is used to read or write the FIFO space for endpoint 2 or channel 2, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE1000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO2D																															

Bit	Name	Reset	Access	Description
31:0	FIFO2D	0xXXXXXXXX	RWH	**Device EP 2/Host Channel 2 FIFO**
	FIFO 2 push/pop region. Used in slave mode.			

38.7.80 USB_FIFO3Dx - Device EP 3/Host Channel 3 FIFO (Actionable Reads)

This register, available in both Host and Device modes, is used to read or write the FIFO space for endpoint 3 or channel 3, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE2000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO3D																															

Bit	Name	Reset	Access	Description
31:0	FIFO3D	0xXXXXXXXX	RWH	**Device EP 3/Host Channel 3 FIFO**
	FIFO 3 push/pop region. Used in slave mode.			

38.7.81 USB_FIFO4Dx - Device EP 4/Host Channel 4 FIFO (Actionable Reads)

This register, available in both Host and Device modes, is used to read or write the FIFO space for endpoint 4 or channel 4, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE3000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO4D																															

Bit	Name	Reset	Access	Description
31:0	FIFO4D	0xXXXXXXXX	RWH	**Device EP 4/Host Channel 4 FIFO**
	FIFO 4 push/pop region. Used in slave mode.			

38.7.82 USB_FIFO5Dx - Device EP 5/Host Channel 5 FIFO (Actionable Reads)

This register, available in both Host and Device modes, is used to read or write the FIFO space for endpoint 5 or channel 5, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE4000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO5D																															

Bit	Name	Reset	Access	Description
31:0	FIFO5D	0xXXXXXXXX	RWH	**Device EP 5/Host Channel 5 FIFO**
	FIFO 5 push/pop region. Used in slave mode.			

38.7.83 USB_FIFO6Dx - Device EP 6/Host Channel 6 FIFO (Actionable Reads)

This register, available in both Host and Device modes, is used to read or write the FIFO space for endpoint 6 or channel 6, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE5000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO6D																															

Bit	Name	Reset	Access	Description
31:0	FIFO6D	0xXXXXXXXX	RWH	**Device EP 6/Host Channel 6 FIFO**
	FIFO 6 push/pop region. Used in slave mode.			

38.7.84 USB_FIFO7Dx - Host Channel 7 FIFO (Actionable Reads)

This register, available in Host mode, is used to read or write the FIFO space for channel 7, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE6000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO7D																															

Bit	Name	Reset	Access	Description
31:0	FIFO7D	0xXXXXXXXX	RWH	**Host Channel 7 FIFO**
	FIFO 7 push/pop region. Used in slave mode.			

38.7.85 USB_FIFO8Dx - Host Channel 8 FIFO (Actionable Reads)

This register, available in Host mode, is used to read or write the FIFO space for channel 8, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE7000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO8D																															

Bit	Name	Reset	Access	Description
31:0	FIFO8D	0xXXXXXXXX	RWH	**Host Channel 8 FIFO**
	FIFO 8 push/pop region. Used in slave mode.			

38.7.86 USB_FIFO9Dx - Host Channel 9 FIFO (Actionable Reads)

This register, available in Host mode, is used to read or write the FIFO space for channel 9, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE8000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO9D																															

Bit	Name	Reset	Access	Description
31:0	FIFO9D	0xXXXXXXXX	RWH	**Host Channel 9 FIFO**
	FIFO 9 push/pop region. Used in slave mode.			

38.7.87 USB_FIFO10Dx - Host Channel 10 FIFO (Actionable Reads)

This register, available in Host mode, is used to read or write the FIFO space for channel 10, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xE9000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO10D																															

Bit	Name	Reset	Access	Description
31:0	FIFO10D	0xXXXXXXXX	RWH	**Host Channel 10 FIFO**
	FIFO 10 push/pop region. Used in slave mode.			

38.7.88 USB_FIFO11Dx - Host Channel 11 FIFO (Actionable Reads)

This register, available in Host mode, is used to read or write the FIFO space for channel 11, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xEA000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO11D																															

Bit	Name	Reset	Access	Description
31:0	FIFO11D	0xXXXXXXXX	RWH	**Host Channel 11 FIFO**
	FIFO 11 push/pop region. Used in slave mode.			

38.7.89 USB_FIFO12Dx - Host Channel 12 FIFO (Actionable Reads)

This register, available in Host mode, is used to read or write the FIFO space for channel 12, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xEB000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO12D																															

Bit	Name	Reset	Access	Description
31:0	FIFO12D	0xXXXXXXXX	RWH	**Host Channel 12 FIFO**
	FIFO 12 push/pop region. Used in slave mode.			

38.7.90 USB_FIFO13Dx - Host Channel 13 FIFO (Actionable Reads)

This register, available in Host mode, is used to read or write the FIFO space for channel 13, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel. This register does not follow dbg_master rule.

Offset	Bit Position																															
0xEC000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFO13D																															

Bit	Name	Reset	Access	Description
31:0	FIFO13D	0xXXXXXXXX	RWH	**Host Channel 13 FIFO**
	FIFO 13 push/pop region. Used in slave mode.			

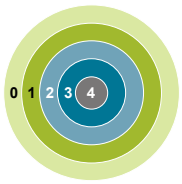
38.7.91 USB_FIFORAMx - Direct Access to Data FIFO RAM for Debugging (2 KB) (Actionable Reads)

This register does not follow dbg_master rule.

Offset	Bit Position																													
0xFE000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0xXXXXXXXX																															
Access	RWH																															
Name	FIFORAM																															

Bit	Name	Reset	Access	Description
31:0	FIFORAM	0xXXXXXXXX	RWH	**FIFO RAM**
	Direct Access to Data FIFO RAM for Debugging (2 KB)			

39. SDIO - SDIO Host controller



Quick Facts

What?

The SDIO Peripheral is a SD/SDIO/MMC host controller with an AHB processor interface, compliant to SD3.01, SDIO3.0 and MMC4.51 standards

Why?

The SDIO host controller handles read or write to memory and peripheral devices

How?

The Host controller has dedicated a DMA which allows to read or write cards without the intervention from the CPU

39.1 Introduction

SDIO is an SD3.01 / SDIO3.0 / eMMC4.51 compliant Host Controller capable of transferring data to SD/MMC/SDIO cards and devices. The module conforms to SD Host Controller Standard Specification Version 3.00.

SDIO is used typically as the interface for other devices, such as a Wi-Fi SoC. SD or MMC are used typically for mass storage - SD is predominantly used for plug-in SD memory cards, while MMC is predominantly used for solder-down eMMC memory ICs. The majority of eMMC and SD memory devices make use of managed NAND flash, which have an integrated flash memory controller that takes care of all wear leveling, error correction, bad block management, etc.

The SDIO Host Controller handles the SDIO/SD/MMC Protocol at the transmission level: packing data, adding the cyclic redundancy check (CRC) bit, driving the Start/End bit, and checking for transaction format correctness. The SDIO Host Controller provides both a Programmed IO (PIO) and a DMA data transfer method.

SDIO documentation is reproduced with permission from Arasan Chip Systems, Inc.

39.2 Features

- Compliance
  - SD Host Controller Standard Specification Version 3.00
  - SDIO card specification Version 3.0
  - SD Memory Card Specification Version 3.01
  - SD Memory Card Security Specification version 1.01
  - MMC Specification version 4.51
- System/Host Interface
  - Data transfer using PIO mode on the Host Bus Slave interface, using DMA mode on the Host Bus Master interface.Host Bus Master interface.
  - AHB master with dedicated DMA controller
  - IP registers and additional registers are configured with AHB slave interface
- SD/SDIO Card Interface
  - Variable host clock rate (supported clock rates can be found in the device datasheet)
  - 1-bit and 4-bit transfers in SD modes
  - Supports SDR104, SDR50, DDR50 modes
  - Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
  - Variable-length data transfers
  - Performs Read wait Control, Suspend/Resume operation SDIO CARD.
  - Designed to work with I/O cards, Read-only cards and Read/Write cards
  - Supports Read wait Control, Suspend/Resume operation
- MMC Card Interface
  - Variable host clock rate (supported clock rates can be found in the device datasheet)
  - 1-bit, 4-bit and 8-bit transfers
  - Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
  - Supports MMC Plus and MMC Mobile
  - Card Detection (Insertion / Removal)
- Miscellaneous
  - SPI mode support
  - Configurable (Minimum 1 Block Size) FIFO used to aid data transfer between the CPU and the Host Controller
  - Handles FIFO Overrun and Underrun conditions by stopping SD_CLK
  - CD and DAT1 pins are overlayed on EM4 wakeup pins, allowing any toggle on CD/DAT1 pins to wakeup the system from EM4

39.3 SDIO Functional Description

39.3.1 Overview

The SDIO block consists of an SDIO core with a DMA controller and AHB slave interface for configuring the core registers (located from 0x400F1000 ~ 0x400F10FF). The SDIO core input's can be configured using another AHB slave interface - these additional registers are located at address location 0x400F1800 ~ 0x400F183F. The SDIO DMA controller as a master transfers data between System memory and Dual port FIFO RAM.

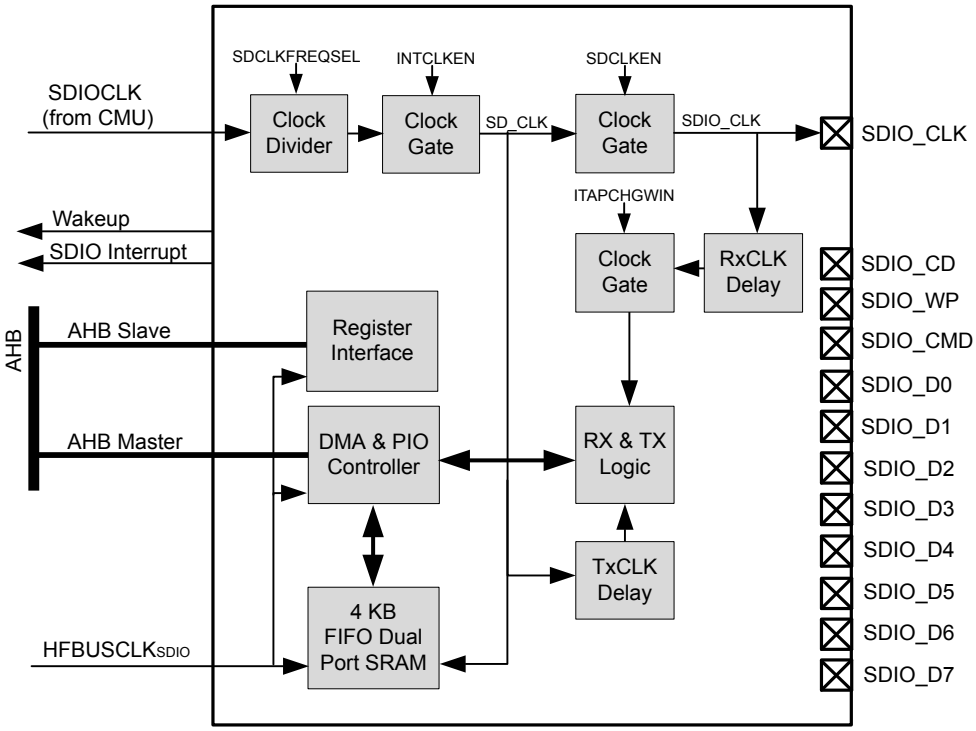


Figure 39.1. SDIO Module Block Diagram

39.3.2 Clocks

- **SDIOCLK -** This is the main clock from the CMU used for the SD Interface side of the Host Controller. SDIOCLK is configured using the SDIOCLKSEL bitfield in the CMU_SDIOCTRL register. This clock is used to generate the SD_CLK, based on the divisor value programmed SDCLKFREQSEL in the SDIO_CLOCKCTRL register.
- **SD_CLK -** This clock is derived from SDIOCLK based on the based on the divisor value programmed SDCLKFREQSEL in SDIO_CLOCKCTRL. SD_CLK is used by the most of the SD Control logic (SD Command Control, SD Transmit Control, SD Tuning Module and one side of the Block Buffer), and is also used to generate the external clock signal to the SD Card (see SDIO_CLK).
- **SDIO_CLK -** Clock signal on the external SDIO_CLK pin supplied to the the SD Card. Same as SD_CLK, except that this clock is available only when SD_CLK Enable is set.
- **$HFBUSCLK_{SDIO}$ -** Host Bus Clock. The Host Bus Interface modules (AHB Master/Slave Modules), the SD Register Set and the DMA Controller operate on this Clock.

39.3.3 Host Interface

The SDIO core interfaces to the System bus using both the AHB Master and Slave Interfaces.

The AHB Slave Bus Interface is used for Host Controller registers accesses, as well as SD Data Port Register accesses when operating in PIO mode.

The AHB Master Bus Interface is used by the DMA Controller (when using DMA or ADMA2 Modes). The DMA Controller uses the AHB Master Bus Interfaces to transfer data between the Dual Port RAM (Block Buffer) and the System Memory, and also to fetch the Descriptors while operating in ADMA2 mode.

39.3.4 Block Buffer

The SDIO block contains a 4KB Dual Port RAM for use as a Block Buffer. The Block Buffer size is configurable, and must be a minimum of 1 block size (where block size is 512 Bytes for SD and up to 2 KBytes for SDIO). For maximum performance, the Block Buffer should be configured for twice the maximum block size (so for SDIO Block Buffer size should be set to 4 KBytes).

One side of the Block Buffer interfaces with the DMA Controller and operates on the $HFBUSCLK_{SDIO}$. The other side of the Block Buffer interfaces with SDIO Control Logic and operates on SD_CLK.

The Block Buffer uses a Circular Buffer Architecture. During a write transaction (e.g., data is transferred from the core to an SD card), the data is fetched from the System Memory and is stored in the Block Buffer. When a block of data is available, the SD Control logic will transfer it onto the SD Interface. The DMA Controller continues to fetch additional blocks of data when the Block Buffer has space. During a read transaction (e.g., data transferred from an SD card to the core), the data from the SD card will be written into the Block Buffer, and after the CRC of the data block is validated, the data will be committed. When a block of data is available, the DMA Controller transfers this data to the System Memory. Provided there is space in the Block Buffer, the SD Control logic will meanwhile receive the next block of data . If the SD Host controller cannot accept any data from the SD card, then it will either issue a read wait (if supported by card) to stop the data transfer from the card, or it will stop the clock on the SDIO_CLK pin.

39.3.5 Interrupts

Table 39.1. SDIO Interrupts

Register	Offset	Description	Access
SDIO_IFCR	0x030	Interrupt status register. One interrupt (IRQ_SDIO) is connected to the EMU	RW1
SDIO_IFENC	0x034	Interrupt status enable register. • If the bit is set then only corresponding event status is set in SDIO_IFCR register • If the bit is cleared, the corresponding bit in SDIO_IFCR is cleared	RW
SDIO_IEN	0x038	Interrupt signal enable register. Interrupt is generated when any of these bits are set and at least one of the status bits in SDIO_IFCR is set to 1	RW

39.3.6 UHS-1 Voltage Switch

GPIOs should be used to configure an external voltage regulator to supply power to the card and the SDIO IOVDD supply to support voltage switching. Software is responsible for configuring the external power supply and setting the correct pin mode in the SDBUSVOLTSEL bitfield in the SDIO_HOSTCTRL1 register.

39.3.7 SDIO Pin Configurations

Table 39.2 SDIO Pins on page 1683 shows different routing options for SDIO pins and also states GPIO mode configuration requirements for each SDIO pins (Please note that GPIO mode configuration is required only when corresponding SDIO pin is enabled via SDIO_ROUTEPEN).

Table 39.2. SDIO Pins

SDIO Pins	LOC0	LOC1	LOC2	LOC3	GPIO MODEn
CLK	E13[1]	E14			PUSHPULL / PUSHPULLALT
CMD	E12[1]	E15			PUSHPULL / PUSHPULLALT
DAT0	E11[1]	A0			PUSHPULL / PUSHPULLALT
DAT1	E10[1,2]	A1[2]			PUSHPULL / PUSHPULLALT
DAT2	E9[1]	A2			PUSHPULL / PUSHPULLALT
DAT3	E8[1]	A3			PUSHPULL / PUSHPULLALT
DAT4	D12[1]	A4			PUSHPULL / PUSHPULLALT
DAT5	D11[1]	A5			PUSHPULL / PUSHPULLALT
DAT6	D10[1]	B3			PUSHPULL / PUSHPULLALT
DAT7	D9[1]	B4			PUSHPULL / PUSHPULLALT
CD	F8[2]	C4[2]	A6[2]	B10[2]	INPUT
WP	F9	C5	B15	B9	INPUT / INPUTPULL

1 High speed location
2 Overlayed onto EM4 wake-up pins

39.3.8 Drivers

The SDIO output drivers support 4 programmable driver strengths. Firmware may select different driver strengths depending on clock frequency and capacitive load.

Table 39.3. SDIO Driver Strength

Driver Type	Nominal Impedance	Maximum Supported Capacitance at 100 MHz SDR or 50 MHz DDR	Notes
A	33 ohm	43 pF	Supports up to 208MHz
B	50 ohm	30 pF	Default driver strength
C	66 ohm	23 pF	Weakest driver that supports up to 208MHz
D	100 ohm	22 pF	For non-speed critical systems, or systems where low-noise is critical.

39.3.9 Pull-Up Resistors

Each of SDIO CMD/DAT lines require a pull-up resistor of 10K~100KOhm when in use.

Figure 39.2 SDIO SD Card BUS Circuitry on page 1684 shows external pull-up resistor connection to different SDIO lines. The pull-up resistor can also be internal as alternative to external. Enabling of the internal on-chip pull-up resistor on a particular SDIO line is controlled by GPIO_Px_DOUT of the corresponding GPIO pin.

Steps to individually enable internal on-chip pull-up resistor for a particular CMD/DAT line are:

1. Select route location for corresponding line in SDIO_ROUTELOC0/SDIO_ROUTELOC1.
2. Configure involved GPIO pin MODEn as PUSHPULL/PUSHPULLALT and set corresponding GPIO_Px_DOUT to '1'.
3. Enable corresponding pin in SDIO_ROUTEPEN.

When external resistor is connected on a particular CMD/DAT line, the required steps are:

1. Select route location for corresponding line in SDIO_ROUTELOC0/SDIO_ROUTELOC1.
2. Configure involved GPIO pin MODEn as PUSHPULL/PUSHPULLALT and set corresponding GPIO_Px_DOUT to '0'.
3. Enable corresponding pin in SDIO_ROUTEPEN.

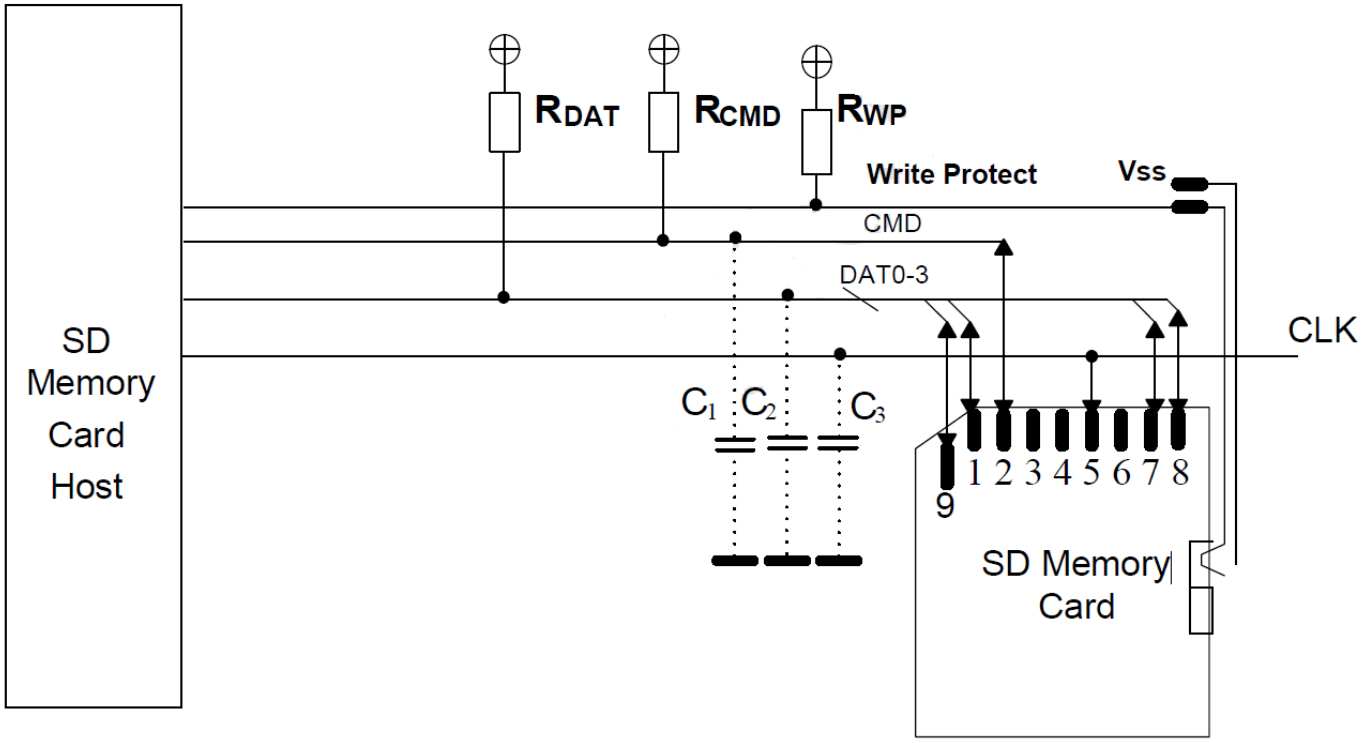


Figure 39.2. SDIO SD Card BUS Circuitry

39.3.10 PIO/DMA Controller

The PIO/DMA Controller Module implements the SDMA and ADMA2 Engines as defined in the SD Host Controller Specification and maintains the block transfer counts for PIO operation. It interacts with the Register Set and starts the DMA Engine when a Command with Data Transfer is involved. The DMA Controller interfaces to the Host (AHB) Master Module to generate transfers, and interfaces with the Block Buffer to store/fetch block data. The DMA Controller implements a separate DMA for SDMA and ADMA2 operation. In addition it implements a Host Transaction Generator that generates controls for the Host Master Interface Module.

39.3.11 SD Transmit Control

The SD Transmit Control Module is used for write data transfers to the Card. Once the Write Command is issued, this module waits for the block of data to be available in the Block Buffer and transfers this onto the external SDIO_DAT bus. Based on the configuration of data lines (1-bit, 4-bit or 8-bits), the data from Block Buffer is appropriately routed.

The CRC16 is individually calculated on per lane basis and is attached at the end of block transfer before the END Bit. In case of DDR operation, it implements a separate CRC16 for each edge of the clock.

At the end of Block transfer, it waits for the CRC Response on the SDIO_DAT0 line and reports the result of the CRC check to the Register Set. Also this module checks for the Write Busy indication (SDIO_DAT0 Line) before transferring next block of data.

A Timeout Check is implemented to make sure that the Write Busy is asserted for no longer than the required limit.

39.3.12 SD Receive Control

The SD Receive Control Module is used for read transfers for receiving data from the Card. Once the Command is issued, this module waits for the block of data to be received from the Card. Based on the configuration of data lines (1-bit, 4-bit or 8-bits), the data from the SD interface is assembled into a byte and eventually into a 32-bit word before being written into the Block Buffer.

The CRC16 is individually calculated on per lane basis and is checked against the received CRC16 at the end of block transfer before the END Bit. In case of DDR operation, it implements a separate CRC16 checker for each edge of the clock.

The data is received on the receive clock, which may be either the looped back external clock (SDIO_CLK) or the Tuned Clock using DLL or DLY elements.

A Timeout Check is implemented to make sure that the gap between the block is no more than the required limit.

39.3.13 SD Card Detect

The SD Card Detect logic monitors the SDIO_CD pin for Card Insertion/Removal events. It implements debouncing logic to filter false transitions on the SDIO_CD Pin. The Card Insertion and Removal events are reported to the SD Host Register set from which the interrupt is eventually generated.

39.3.14 FIFO Overrun and Underrun Conditions in DMA and Non-DMA Transfers

Write

During write transaction, the Host Controller will transmit data to a SD card only when a block of data is ready to transmit and also card is not driving busy. So an underrun condition will not occur on the SD side. In DMA mode, the Host Controller initiates a DMA READ from Host Memory only if space is available to accept a block of data. In non-DMA mode, the Host Controller will assert the buffer write ready interrupt only if space is available to accept a block of data.

Read

When the internal Block Buffer is full during a read transaction, it can not accept any more data from the card. In this case, the Host Controller will stop the clock to the SD card in order to avoid a Buffer Overrun condition. In SDIO mode with Read wait enabled, the Read Wait signal is asserted to block the Card from sending any more data. In DMA mode, the Host Controller initiates a DMA WRITE to Host Memory only on reception of block of data from card.

In non-DMA mode, the Host Controller will assert the buffer read ready interrupt only on receipt of a block of data from the card.

39.3.15 Supported Card Sizes

SD/MMC/SDIO memory cards up to 2 TB can be supported. In ADMA2 DMA mode, the maximum block count is set by a 32-bit descriptor, and in SDMA mode or non-DMA mode, block count is set by a 16-bit register. For SD/MMC/SDIO memory cards, each sector size (block size) is 512 bytes, resulting in a total card capacity of 2^{32} x 512 Bytes (i.e., 2 TBytes).

39.4 Register Map

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

Offset	Name	Type	Description
0x000	SDIO_SDMASYSADDR	RW	SDMA System Address Register
0x004	SDIO_BLKSIZE	RW	Block Size and Block Count Register
0x008	SDIO_CMDARG1	RW	SD Command Argument Register
0x00C	SDIO_TFRMODE	RW	Transfer Mode and Command Register
0x010	SDIO_RESP0	R	Response0 and Response1 Register
0x014	SDIO_RESP2	R	Response2 and Response3 Register
0x018	SDIO_RESP4	R	Response4 and Response5 Register
0x01C	SDIO_RESP6	R	Response6 and Response7 Register
0x020	SDIO_BUFDATPORT	RW	Buffer Data Register
0x024	SDIO_PRSSTAT	R	Present State Register
0x028	SDIO_HOSTCTRL1	RWH	Host Control1, Power, Block Gap and Wakeup-up Control Register
0x02C	SDIO_CLOCKCTRL	RWH	Clock Control, Timeout Control and Software Register
0x030	SDIO_IFCR	RWH	Normal and Error Interrupt Status Register
0x034	SDIO_IFENC	RW	Normal and Error Interrupt Status Enable Register
0x038	SDIO_IEN	RW	Normal and Error Interrupt Signal Enable Register
0x03C	SDIO_AC12ERRSTAT	RWH	AUTO CMD12 Error Status and Host Control2 Register
0x040	SDIO_CAPAB0	R	Capabilities Register to Hold Bits 31~0
0x044	SDIO_CAPAB2	R	Capabilities Register to Hold Bits 63~32
0x048	SDIO_MAXCURCAPAB	R	Maximum Current Capabilities Register
0x050	SDIO_FEVTERRSTAT	RWH	Force Event Register for Auto CMD Error Status
0x054	SDIO_ADMAES	R	ADMA Error Status Register
0x058	SDIO_ADSADDR	RW	ADMA System Address Register
0x060	SDIO_PRSTVAL0	R	Preset Value for Initialization and Default Speed Mode
0x064	SDIO_PRSTVAL2	R	Preset Value for High Speed and SDR12 Modes
0x068	SDIO_PRSTVAL4	R	Preset Value for SDR25 and SDR50 Modes
0x06C	SDIO_PRSTVAL6	R	Preset Value for SDR104 and DDR50 Modes
0x070	SDIO_BOOTTOCTRL	RW	Boot Timeout Control Register
0x0FC	SDIO_SLOTINTSTAT	R	Slot Interrupt Status Register
0x800	SDIO_CTRL	RW	Core Control Signals
0x804	SDIO_CFG0	RW	Core Configuration 0
0x808	SDIO_CFG1	RW	Core Configuration 1
0x80C	SDIO_CFGPRESETVAL0	RW	Core Configuration Preset Value 0
0x810	SDIO_CFGPRESETVAL1	RW	Core Configuration Preset Value 1
0x814	SDIO_CFGPRESETVAL2	RW	Core Configuration Preset Value 2
0x818	SDIO_CFGPRESETVAL3	RW	Core Configuration Preset Value 3

Offset	Name	Type	Description
0x81C	SDIO_ROUTELOC0	RW	I/O LOCATION Register
0x820	SDIO_ROUTELOC1	RW	I/O LOCATION Register
0x824	SDIO_ROUTEPEN	RW	I/O LOCATION Enable Register

39.5 Register Description

39.5.1 SDIO_SDMASYSADDR - SDMA System Address 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	0x00000000																															
Access	RW																															
Name	SDMASYSADDRARG																															

Bit	Name	Reset	Access	Description
31:0	SDMASYSAD-DRARG	0x00000000	RW	**Physical SYS Memory ADDR Used for DMA Transfers or the Second Argument for the Auto CMD23**
	Contains (1) SDMASYSADDR : This register contains the SYS memory ADDR for a SDMA transfer. (2) ARGUMENT2 : This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23.			

39.5.2 SDIO_BLKSIZE - Block Size and Block Count 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	0x0000																	0x0			0x000											
Access	RW																	RW			RW											
Name	BLKSCNTFORCURRTFR																	HSTSDMABUFSIZE			TFRBLKSIZE											

Bit	Name	Reset	Access	Description
31:16	BLKSCNTFOR-CURRTFR	0x0000	RW	**Blocks Count for Current Transfer**

This register is enabled when Block Count Enable in the Tranfer Mode register is set to 1 and is valid only for multiple block transfers. 0000h - Stop Count 0001h - 1 block 0002h - 2 blocks FFFFh - 65535 blocks

Value	Description
BLKSCNTFOR-CURRTFR	Block count of BLKSCNTFOR-CURRTFR.

15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14:12	HSTSDMABUFSIZE	0x0	RW	**Host SDMA Buffer Size**

These bits specify the size of contiguous buffer in the SYS memory

Value	Mode	Description
0	SIZE4	4KB(Detects A11 Carry out)
1	SIZE8	8KB(Detects A12 Carry out)
2	SIZE16	16KB(Detects A13 Carry out)
3	SIZE32	32KB(Detects A14 Carry out)
4	SIZE64	64KB(Detects A15 Carry out)
5	SIZE128	128KB(Detects A16 Carry out)
6	SIZE256	256KB(Detects A17 Carry out)
7	SIZE512	512KB(Detects A18 Carry out)

11:0	TFRBLKSIZE	0x000	RW	**Transfer Block Size, Specifies the Block Size for Block Data Transfers for CMD17, CMD18, CMD24, CMD25, and CMD53**

It can be accessed only if no transaction is executing (i.e after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored.

Value	Description

Bit	Name	Reset	Access	Description
	TRFBLKSIZE	Transfer block size of TRFBLKSIZE.		

39.5.3 SDIO_CMDARG1 - SD Command Argument 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	0x00000000																															
Access	RW																															
Name	CMDARG1																															

Bit	Name	Reset	Access	Description
31:0	CMDARG1	0x00000000	RW	**Command Argument 1**
	The SD CMD Argument is specified as bits 39-8 of Command-Format.			

39.5.4 SDIO_TFRMODE - Transfer Mode and 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			0x00						0x0		0	0	0		0x0												0	0	0x0		0	0
Access			RW						RW		RW	RW	RW		RW												RW	RW	RW		RW	RW
Name			CMDINDEX						CMDTYPE		DATPRESSEL	CMDINDXCHKEN	CMDCRCCHKEN		RESPTYPESEL												MULTSINGBLKSEL	DATDIRSEL	AUTOCMDEN		BLKCNTEN	DMAEN

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	CMDINDEX	0x00	RW	**Command Index**
	This bit shall be set to the command number (CMD0-63, ACMD0- 63).			
23:22	CMDTYPE	0x0	RW	**Command Type**
	There are three types of special CMDs. Suspend, Resume and Abort. These bits shall be set to 00b for all other CMDs.			

Value	Mode	Description
0	NORMAL	Normal Command
1	SUSPEND	Suspend command
2	RESUME	Resume command
3	ABORT	Abort command

Bit	Name	Reset	Access	Description
21	DATPRESSEL	0	RW	**Data Present Select**
	This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. It is set to 0 for the following: 1. CMDs using only CMD line (ex. CMD52) 2. CMDs with no data transfer but using busy signal on DAT[0] line (R1b or R5b ex. CMD38)			

Value	Mode	Description
0	NODATA	No Data present
1	DATA	Data present

Bit	Name	Reset	Access	Description
20	CMDINDXCHKEN	0	RW	**Command Index Check Enable**
	If this bit is set to 1, the host controller shall check the index field in the RESP to see if it has the same value as the CMD index. If it is not, it is reported as a CMD Index ERR. If this bit is set to 0, the Index field is not checked			

Value	Mode	Description
0	DISABLE	Disable
1	ENABLE	Enable

Bit	Name	Reset	Access	Description
19	CMDCRCCHKEN	0	RW	**Command CRC Check Enable**

If this bit is set to 1, the host controller shall check the CRC field in the RESP. If an error is detected, it is reported as a CMD CRC ERR. If this bit is set to 0, the CRC field is not checked

Value	Mode	Description
0	DISABLE	Disable
1	ENABLE	Enable

18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
17:16	RESPTYPESEL	0x0	RW	**Response Type Select**

Response Type Selection

Value	Mode	Description
0	NORESP	No RESP
1	RESP136	RESP Length 136
2	RESP48	RESP Length 48
3	BUSYAFTRESP	RESP Length 48 check busy after RESP

15:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	MULTSINGBLKSEL	0	RW	**Multiple or Single Block Data Transfer Selection**

This bit enables multiple block data transfers.

Value	Mode	Description
0	SINGLEBLK	Single block data transfer
1	MULTIBLK	Multi block data transfer

4	DATDIRSEL	0	RW	**Data Transfer Direction Select**

This bit defines the direction of data transfers.

Value	Mode	Description
0	DISABLE	Disable
1	ENABLE	Enable

3:2	AUTOCMDEN	0x0	RW	**Auto Command Enable**

This field determines use of auto CMD functions.

Value	Mode	Description
0	ACMDDISABLED	Auto CMD Disabled
1	ACMD12EN	Auto CMD12 Enable
2	ACMD23EN	Auto CMD23 Enable

1	BLKCNTEN	0	RW	**Block Count Enable**

This bit is used to enable the Block count register, which is only relevant for multiple block transfers.

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	DISABLE		Disable
	1	ENABLE		Enable
0	DMAEN	0	RW	**DMA Enable**
	DMA can be enabled only if DMA Support bit in the Capabilities register is set.			
	Value	Mode		Description
	0	DISABLE		Disable
	1	ENABLE		Enable

39.5.5 SDIO_RESP0 - Response0 and Response1 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	0x00000000																															
Access	R																															
Name	CMDRESP0																															

Bit	Name	Reset	Access	Description
31:0	CMDRESP0	0x00000000	R	**Command Response 0**
	Command response is either short response 32-bits or long response 128-bits. This register shows either short response bits [31:0] or long response bits [127:96]			

39.5.6 SDIO_RESP2 - Response2 and Response3 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	0x00000000																															
Access	R																															
Name	CMDRESP1																															

Bit	Name	Reset	Access	Description
31:0	CMDRESP1	0x00000000	R	**Command Response 1**
	This register shows long response bits [95:64]			

39.5.7 SDIO_RESP4 - Response4 and Response5 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	0x00000000																															
Access	R																															
Name	CMDRESP2																															

Bit	Name	Reset	Access	Description
31:0	CMDRESP2	0x00000000	R	**Command Response 2**
	This register shows long response bits [63:32]			

39.5.8 SDIO_RESP6 - Response6 and Response7 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	0x00000000																															
Access	R																															
Name	CMDRESP3																															

Bit	Name	Reset	Access	Description
31:0	CMDRESP3	0x00000000	R	**Command Response 3**
	This register shows long response bits [31:0]			

39.5.9 SDIO_BUFDATPORT - Buffer 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	0x00000000																															
Access	RW																															
Name	BUFDAT																															

Bit	Name	Reset	Access	Description
31:0	BUFDAT	0x00000000	RW	**Buffer Data**
	The Host Controller Buffer can be accessed through this 32-bit Data Port Register.			

39.5.10 SDIO_PRSSTAT - Present State 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				0	0x0				0	0	0	0					0	0	0	0					0	0	0	0
Access				R				R	R				R	R	R	R					R	R	R	R					R	R	R	R
Name				DAT7TO4SIGLVL				CMDSIGLVL	DAT3TO0SIGLVL				WRPROTSWPINLVL	CARDDETPINLVL	CARDSTATESTABLE	CARDINS					BUFRDEN	BUFFERWRITEENABLE	RDTRANACT	WRTRANACT					RETUNINGREQ	DATLINEACTIVE	CMDINHIBITDAT	CMDINHIBITCMD

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*		
28:25	DAT7TO4SIGLVL	0x0	R	**DAT[7:4] Line Signal Level** This status bit is used to check DAT line level to recover from errors, and for debugging. D28 - DAT[7] D27 - DAT[6] D26 - DAT[5] D25 - DAT[4]
24	CMDSIGLVL	0	R	**Command Line Signal Level** This status bit is used to check CMD line level to recover from errors, and for debugging.
23:20	DAT3TO0SIGLVL	0x0	R	**DAT[3:0] Line Signal Level** This status bit is used to check DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DAT[0]. D23 - DAT[3] D22 - DAT[2] D21 - DAT[1] D20 - DAT[0]
19	WRPROTSWPINLVL	0	R	**Write Protect Switch Pin Level** The Write Protect Switch is supported for memory and combo cards. This bit reflects the SDIO_WP pin. 0 - Write protected (SDIO_WP = 1) 1 - Write enabled (SDIO_WP = 0)
18	CARDDETPINLVL	0	R	**Card Detect Pin Level** This bit reflects the inverse value of the SDIO_CD pin. 0 - No Card present (SDIO_CD = 1) 1 - Card present (SDIO_CD = 0)
17	CARDSTATESTABLE	0	R	**Card State Stable Status** This bit is used for testing. If it is 0, the Card Detect (SDIO_CD) Pin Level is not stable. If this bit is set to 1, it means the Card Detect (SDIO_CD) Pin Level is stable.
16	CARDINS	0	R	**Card Inserted Status** This bit indicates whether a card has been inserted. Changing from 0 to 1 generates a Card Insertion INT in the Normal INT STAT register and changing from 1 to 0 generates a Card Removal INT in the Normal INT STAT register.
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11	BUFRDEN	0	R	**Buffer Read Enable** This status bit is used for non-DMA read transfers. This read only flag indicates that valid data exists in the host side buffer status bit.

Bit	Name	Reset	Access	Description
10	BUFFERWRITEENABLE	0	R	**Buffer Write Enable** This status bit is used for non-DMA write transfers. This read only flag indicates if space is available for write data.
9	RDTRANACT	0	R	**Read Transfer Active** This status bit is used for detecting completion of a read transfer.
8	WRTRANACT	0	R	**Write Transfer Active** This status bit indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the host controller.
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	RETUNINGREQ	0	R	**Re-Tuning Request** Re-Tuning Request Host Controller may request Host Driver to execute re-tuning sequence by setting this bit when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data.
2	DATLINEACTIVE	0	R	**DAT Line Active** This bit indicates whether one of the DAT line on SD bus is in use. 1 - DAT line active 0 - DAT line inactive
1	CMDINHIBITDAT	0	R	**Command Inhibit (DAT)** This status bit is generated if either the DAT Line Active or the Read transfer Active is set to 1. If this bit is 0, it indicates the host controller can issue the next SD CMD.
0	CMDINHIBITCMD	0	R	**Command Inhibit (CMD)** If this bit is 0, it indicates the CMD line is not in use and the host controller can issue a SD CMD using the CMD line.

39.5.11 SDIO_HOSTCTRL1 - Host Control1, Power, Block Gap and Wakeup-up 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						0	0	0	1	0	0	0	0	0	0	0				0	0x0			0	0	0	0	0x0		0	0	0
Access						RW	RW	RW	RW	RW	RW	RW	RW	RW	RWH	RW				RW	RW			RW	RW	RW	RW	RW		RW	RW	RW
Name						WKUPEVNTENONCRM	WKUPEVNTENONCINS	WKUPEVNTENONCARDINT	BOOTACKCHK	ALTBOOTEN	BOOTEN	SPIMODE	INTATBLKGAP	RDWAITCTRL	CONTINUEREQ	STOPATBLKGAPREQ				HRDRST	SDBUSVOLTSEL			SDBUSPOWER	CDSIGDET	CDTSTLVL	EXTDATTRANWD	DMASEL		HSEN	DATTRANWD	LEDCTRL

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	WKUPEVNTENONCRM	0	RW	**Wakeup Event Enable on SD Card Removal**
	This bit enables wakeup event via Card Removal assertion in the Normal INT STAT register. FN_WUS (Wake up Support) in CIS does not affect this bit. 1 - Enable 0 - Disable			
25	WKUPEVNTENONCINS	0	RW	**Wakeup Event Enable on SD Card Insertion**
	This bit enables wakeup event via Card Insertion assertion in the Normal INT STAT register. FN_WUS (Wake up Support) in CIS does not affect this bit. 1 - Enable 0 - Disable			
24	WKUPEVNTENONCARDINT	0	RW	**Wakeup Event Enable on Card Interrupt**
	This bit enables wakeup event via Card INT assertion in the Normal INT STAT register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. 1 - Enable 0 - Disable			
23	BOOTACKCHK	1	RW	**Boot Ack Check**
	To check for the boot acknowledge in boot operation. 1 - wait for boot ack from eMMC card 0 - Will not wait for boot ack from eMMC card			
22	ALTBOOTEN	0	RW	**Alternate Boot Enable**
	To start boot code access in alternative mode. 1- To start alternate boot mode access 0 - To stop alternate boot mode access			
21	BOOTEN	0	RW	**Boot Enable**
	To start boot code access 1- To start boot code access 0 - To stop boot code access			
20	SPIMODE	0	RW	**SPI Mode Enable**
	SPI mode enable bit. 1- SPI mode 0- SD mode			
19	INTATBLKGAP	0	RW	**Interrupt at Block Gap**
	This bit is valid only in 4-bit mode of the SDIO card and selects a sample point in the INT cycle. Setting to 1 enables INT detection at the block gap for a multiple block transfer. If the SD card cannot signal an INT during a multiple block transfer, this bit should be set to 0.			

Bit	Name	Reset	Access	Description
18	RDWAITCTRL	0	RW	**Read Wait Control**
	The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using DAT[2] line. Otherwise the host controller has to stop the SD_CLK to hold read data, which restricts CMDs generation.			
17	CONTINUEREQ	0	RWH	**Continue Request**
	This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request.			
16	STOPATBLKGAP-REQ	0	RW	**Stop at Block Gap Request**
	This bit is used to stop executing a transaction at the next block gap for noDMA,SDMA and ADMA transfers. Until the transfer complete is set to 1, indicating a transfer completion the host driver shall leave this bit set to 1.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12	HRDRST	0	RW	**Hardware Reset Signal**
	Hardware reset signal is generated for eMMC card when this bit is set '1' - Drives the hardware reset pin as ZERO (Active LOW to eMMC card) '0' - Deassert the hardware reset pin			
11:9	SDBUSVOLTSEL	0x0	RW	**SD Bus Voltage Select**
	By setting these bits, the host driver selects the voltage mode for the SD card pins. Changing SDBUSVOLTSEL does not change the actual I/O voltage at the pins. All IOVDD supply voltage switching must be handled externally.			

Value	Mode	Description
5	1P8V	Select 1.8V
6	3P0V	Select 3.0V
7	3P3V	Select 3.3V

8	SDBUSPOWER	0	RW	**SD Bus Power**
	Before setting this bit, the SD host driver shall set SD Bus Voltage SEL. If the host controller detects the No Card State, this bit shall be cleared. 1 - Power on 0 - Power off			
7	CDSIGDET	0	RW	**Card Detetct Signal Detection**
	This bit selects source for card detection.			

Value	Mode	Description
0	SDCD	SDCD selected
1	TSTLVL	Card detect test level selected

6	CDTSTLVL	0	RW	**Card Detect Test Level**
	This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not. Generates (card ins or card removal) INT when the normal int sts enable bit is set.			

Value	Mode	Description
0	NOCARD	No Card
1	CARDIN	Card Inserted

5	EXTDATTRANWD	0	RW	**Extended Data Transfer Width**
	This bit controls 8-bit bus width mode for embedded device. Support of this function is indicated in 8-bit Support for Embedded Device in the Capabilities register.			

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	8BIT		8-bit bus width
	1	DATXFR		Bus width selected by Data Transfer Width
4:3	DMASEL	0x0	RW	**DMA Select**
	One of supported DMA modes can be selected. The host driver shall check support of DMA modes by referring the Capabilities register.			
	Value	Mode		Description
	0	SDMA		SDMA selected
	1	ADMA1		32-bit ADMA1 selected
	2	ADMA2		32-bit ADMA2 selected
	3	64BITADMA2		64-bit ADMA2 selected
2	HSEN	0	RW	**High Speed Enable**
	If this bit is set to 0 (default), the host controller outputs CMD line and DAT lines at the falling edge of the SD_CLK. If this bit is set to 1, the host controller outputs CMD line and DAT lines at the rising edge of the SD_CLK (This bit is optional. Before setting this bit, the host driver shall check the High Speed Support in the capabilities register.)			
	Value	Mode		Description
	0	NS		Normal Speed Mode
	1	HS		High Speed Mode
1	DATTRANWD	0	RW	**Data Transfer Width 1-bit or 4-bit Mode**
	This bit selects the data width of the host controller. The host driver shall select it to match the data width of the SD card. (SD1 or SD4) 1 - 4 bit mode 0 - 1 bit mode			
	Value	Mode		Description
	0	SD1		1-bit mode
	1	SD4		4-bit mode
0	LEDCTRL	0	RW	**LED Control**
	This bit is used to caution the user not to remove the card while the SD card is being accessed. If the software is going to issue multiple SD CMDs, this bit can be set during all transactions. It is not necessary to change for each tranaction.			
	Value	Mode		Description
	0	LEDOFF		LED is OFF
	1	LEDON		LED is ON

39.5.12 SDIO_CLOCKCTRL - Clock Control, Timeout Control and Software 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						0	0	0					0x0				0x00								0x0		0			0	0	0
Access						RWH	RWH	RWH					RW				RW								RW		RW			RW	R	RW
Name						SFTRSTDAT	SFTRSTCMD	SFTRSTA					DATTOUTCNTVAL				SDCLKFREQSEL								UPPSDCLKFRE		CLKGENSEL			SDCLKEN	INTCLKSTABLE	INTCLKEN

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	SFTRSTDAT	0	RWH	**Software Reset for DAT Line**
	Only part of data circuit is reset. The following registers and bits are cleared by this bit: Buffer Data Port Register Buffer is cleared and Initialized. Present State register Buffer read Enable Buffer write Enable Read Transfer Active Write Transfer Active DAT Line Active CMD Inhibit (DAT) Block Gap CTRL register Continue Request Stop At Block Gap Request Normal INT STAT register Buffer Read Ready Buffer Write Ready Block Gap Event Transfer Complete 1 - Reset 0 - Work			
25	SFTRSTCMD	0	RWH	**Software Reset for CMD Line**
	Only part of CMD circuit is reset. The following registers and bits are cleared by this bit: Present State register CMD Inhibit (CMD) Normal INT STAT register CMD Complete 1 - Reset 0 - Work			
24	SFTRSTA	0	RWH	**Software Reset for All**
	This reset affects the entire host controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared to 0.			
23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	DATTOUTCNTVAL	0x0	RW	**Data Timeout Counter Value**
	This value determines the interval by which DAT line time-outs are detected.			
15:8	SDCLKFREQSEL	0x00	RW	**SD_CLK Frequency Select**
	This register is used to select the frequency of the clock output on the SDIO_CLK pin. The frequency is not programmed directly; rather this register holds the divisor of the Base Clock Frequency For SD_CLK in the capabilities register.			

Value	Description
SDCLKFREQSEL	Clock division factor of SDCLKFREQSEL+1.

7:6	UPPSDCLKFRE	0x0	RW	**Upper Bits of SD_CLK Frequency Select**
	Bit 07-06 is assigned to bit 09-08 of clock divider in SD_CLK Frequency SEL.			
5	CLKGENSEL	0	RW	**Clock Generator Select**
	This bit is used to select the clock generator mode in SD_CLK Frequency SEL.			
4:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
2	SDCLKEN	0	RW	**SDIO_CLK Pin Clock Enable**
	The host controller shall stop the clock output on the SDIO_CLK pin when writing this bit to 0. The SD_CLK frequency SEL can be changed when this bit is 0. Then, the host controller shall maintain the same clock frequency until SD_CLK is stopped (i.e., SD_CLK = 0). If the host controller detects the 'No Card' state, this bit shall be cleared. 1 - Enable 0 - Disable			
1	INTCLKSTABLE	0	R	**Internal Clock Stable**
	This bit is set to 1 when SD_CLK is stable after writing to Internal Clock Enable in this register to 1. The SD Host Driver shall wait to set SDCLKEN until this bit is set to 1.			
0	INTCLKEN	0	RW	**Internal Clock Enable**
	This bit is set to 0 when the host driver is not using the host controller or the host controller awaits a wakeup event. The host controller should stop its internal clock to go very low power state. Still, registers shall be able to be read and written. Clock starts to oscillate when this bit is set to 1.			

39.5.13 SDIO_IFCR - Normal and Error Interrupt 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				0			0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0
Access				RW1			RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	R	RW1	RW1	R				R	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name				TARGETRESP			ADMAERR	AUTOCMDERR	CURRENTLIMITERR	DATENDBITERR	DATCRCERR	DATTOUTERR	CMDINDEXERR	CMDENDBITERR	CMDCRCERR	CMDTOUTERR	ERRINT	BOOTTERMINATE	BOOTACKRCV	RETUNINGEVT				CARDINT	CARDRM	CARDINS	BFRRDRDY	BFRWRRDY	DMAINT	BLKGAPEVT	TRANCOM	CMDCOM

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	TARGETRESP	0	RW1	**Specific Error STAT**
	Occurs when detecting a DMA transaction error in error 0 - no error 1 - error			
27:26	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
25	ADMAERR	0	RW1	**ADMA Error**
	This bit is set when the Host controller detects errors during ADMA based data transfer. The state of the ADMA at an error occurrence is saved in the ADMA ERR STAT Register. 1- error 0 -No error			
24	AUTOCMDERR	0	RW1	**Auto CMD Error**
	Auto CMD12 and Auto CMD23 use this ERR STAT. This bit is set when detecting that one of the bits D00-D04 in Auto CMD ERR STAT register has changed from 0 to 1. In case of Auto CMD12, this bit is set to 1, not only when the errors in Auto CMD12 occur but also when Auto CMD12 is not executed due to the previous CMD error. 0 - No error 1 - error			
23	CURRENTLIMITERR	0	RW1	**Current Limit Error**
	By setting the SD Bus Power bit in the Power Control Register, the host controller is requested to supply power for the SD Bus. If the host controller supports the Current Limit Function, it can be protected from an Illegal card by stopping power supply to the card in which case this bit indicates a failure STAT. Reading 1 means the host controller is not supplying power to SD card due to some failure. Reading 0 means that the host controller is supplying power and no error has occurred. This bit shall always set to be 0, if the host controller does not support this function. 0 - No error 1 - Power Fail			
22	DATENDBITERR	0	RW1	**Data End Bit Error**
	Occurs when detecting 0 at the end bit position of read data which uses the DAT line or the end bit position of the CRC STAT. 0 - No error 1 - error			
21	DATCRCERR	0	RW1	**Data CRC Error**
	Occurs when detecting CRC ERR when transferring read data which uses the DAT line or when detecting the Write CRC STAT having a value of other than 010. 0 - No error 1 - error			
20	DATTOUTERR	0	RW1	**Data Time-out Error**
	Occurs when detecting one of following timeout conditions. 1. Busy Timeout for R1b, R5b type. 2. Busy Timeout after Write CRC STAT 3. Write CRC STAT Timeout 4. Read Data Timeout 0 - No error 1 - Timeout			
19	CMDINDEXERR	0	RW1	**Command Index Error**
	Occurs if a CMD Index ERR occurs in the CMD RESP. 0 - No error 1 - error			

Bit	Name	Reset	Access	Description
18	CMDENDBITERR	0	RW1	**Command End Bit Error**
	Occurs when detecting that the end bit of a command RESP is 0. 0 - No error 1 - End Bit error Generated			
17	CMDCRCERR	0	RW1	**CMD CRC Error**
	CMD CRC ERR is generated in two cases. 1. If a RESP is returned and the CMD Time-out ERR is set to 0, this bit is set to 1 when detecting a CRT error in the CMD RESP 2. The host controller detects a CMD line conflict by monitoing the CMD line when a CMD is issued. If the host controller drives the CMD line to 1 level, but detects 0 level on the CMD line at the next SD_CLK edge, then the host controller shall abort the command (Stop driving CMD line) and set this bit to 1. The CMD Timeout error shall also be set to 1 to distinguish CMD line conflict. 0 - No error 1 - CRC error Generated			
16	CMDTOUTERR	0	RW1	**Command Timeout Error**
	Occurs only if the no response is returned within 64 SD_CLK cycles from the end bit of the command. If the host controller detects a CMD line conflict, in which case Command CRC Error shall also be set. This bit shall be set without waiting for 64 SD_CLK cycles because the command will be aborted by the host controller. 0- No Error, 1-Timeout.			
15	ERRINT	0	R	**Error Interrupt**
	If any of the bits in the ERR INT STAT Register are set, then this bit is set. Therefore the host driver can test for an error by checking this bit first. 0 - No error. 1 - error.			
14	BOOTTERMINATE	0	RW1	**Boot Terminate Interrupt**
	This status bit is set if the boot operation get terminated INT 0 - Boot operation is not terminated. 1 - Boot operation is terminated			
13	BOOTACKRCV	0	RW1	**Boot Ack Received**
	This status bit is set if the boot acknowledge is received from device. 0 - Boot ack is not received. 1 - Boot ack is received.			
12	RETUNINGEVT	0	R	**Re-Tunning Event**
	This status bit is set if Re-Tuning Request in the Present State register changes from 0 to 1. Host controller requests Host Driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning. 1 Re-Tuning should be performed 0 Re-Tuning is not required			
11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	CARDINT	0	R	**Card Interrupt**
	Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card INT factor.			
7	CARDRM	0	RW1	**Card Removal**
	This status bit is set if the Card Inserted in the Present State register changes from 1 to 0. When the host driver writes this bit to 1 to clear this status bit, the status of the Card Inserted in the Present State register should be confirmed. Because the card detect may possibly be changed when the host driver clear this bit an Interupt event may not be generated. 0 - Card State Stable or Debouncing 1 - Card Removed			
6	CARDINS	0	RW1	**Card Insertion**
	This status bit is set if the Card Inserted in the Present State register changes from 0 to 1. When the host driver writes this bit to 1 to clear this status bit, the status of the Card Inserted in the Present State register should be confirmed. Because the card detect may possibly be changed when the host driver clear this bit an Interupt event may not be generated. 0 - Card State Stable or Debouncing 1 - Card Inserted			
5	BFRRDRDY	0	RW1	**Buffer Read Ready**
	This status bit is set if the Buffer Read Enable changes from 0 to 1. Buffer Read Ready is set to 1 for every CMD19 execution in tuning procedure. 0 - Not Ready to read Buffer. 1 - Ready to read Buffer.			
4	BFRWRRDY	0	RW1	**Buffer Write Ready**
	This status bit is set if the Buffer Write Enable changes from 0 to 1. 0 - Not Ready to Write Buffer. 1 - Ready to Write Buffer.			

Bit	Name	Reset	Access	Description
3	DMAINT	0	RW1	**DMA Interrupt** This status bit is set if the host controller detects the Host DMA Buffer Boundary in the Block Size register. 0 - No DMA INT 1 - DMA INT is Generated
2	BLKGAPEVT	0	RW1	**Block Gap Event** If the Stop At Block Gap Request in the Block Gap CTRL Register is set, this bit is set.
1	TRANCOM	0	RW1	**Transfer Complete** This bit is set when a read / write transaction is completed.
0	CMDCOM	0	RW1	**Command Complete** This bit is set when we get the end bit of the CMD RESP (Except Auto CMD12 and Auto CMD23) Note: CMD Time-out error has higher priority than CMD Complete. If both are set to 1, it can be considered that the RESP was not received correctly.

39.5.14 SDIO_IFENC - Normal and Error Interrupt Status Enable 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				0		0	0	0	0	0	0	0	0	0	0	0		0	0	0				0	0	0	0	0	0	0	0	0
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				TARGETRESPEN		TUNINGERREN	ADMAERREN	AUTOCMDERREN	CURRENTLIMITERREN	DATENDBITERREN	DATCRCERREN	DATTOUTERREN	CMDINDEXERREN	CMDENDBITERREN	CMDCRCERREN	CMDTOUTERREN		BOOTTERMINATEEN	BOOTACKRCVEN	RETUNINGEVTEN				CARDINTEN	CARDRMEN	CARDINSEN	BUFRDRDYEN	BUFWRRDYEN	DMAINTEN	BLKGAPEVTEN	TRANCOMEN	CMDCOMEN

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	TARGETRESPEN	0	RW	**Target Response/Host Error Status Enable**
	0 - Masked 1 - Enabled			
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	TUNINGERREN	0	RW	**Tuning Error Status Enable**
	0 - Masked 1 - Enabled			
25	ADMAERREN	0	RW	**ADMA Error Status Enable**
	0 - Masked 1 - Enabled			
24	AUTOCMDERREN	0	RW	**Auto CMD12 Error Status Enable**
	0 - Masked 1 - Enabled			
23	CURRENTLIMITERREN	0	RW	**Current Limit Error Status Enable**
	0 - Masked 1 - Enabled			
22	DATENDBITERREN	0	RW	**Data End Bit Error Status Enable**
	0 - Masked 1 - Enabled			
21	DATCRCERREN	0	RW	**Data CRC Error Status Enable**
	0 - Masked 1 - Enabled			
20	DATTOUTERREN	0	RW	**Data Timeout Error Status Enable**
	0 - Masked 1 - Enabled			
19	CMDINDEXERREN	0	RW	**Command Index Error Status Enable**
	0 - Masked 1 - Enabled			
18	CMDENDBITERREN	0	RW	**Command End Bit Error Status Enable**
	0 - Masked 1 - Enabled			
17	CMDCRCERREN	0	RW	**Command CRC Error Status Enable**
	0 - Masked 1 - Enabled			

Bit	Name	Reset	Access	Description
16	CMDTOUTERREN	0	RW	**Command Time-out Error Status Enable** 0 - Masked 1 - Enabled
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	BOOTTERMINATEEN	0	RW	**Boot Terminate Interrupt Signal Enable** 0 - Masked 1 - Enabled
13	BOOTACKRCVEN	0	RW	**Boot Ack Received Signal Enable** 0 - Masked 1 - Enabled
12	RETUNINGEVTEN	0	RW	**Re-Tunning Event Signal Enable** 0 - Masked 1 - Enabled
11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	CARDINTEN	0	RW	**Card Interrupt Signal Enable** If this bit is set to 0, the host controller shall clear INT request to the SYS. The Card INT detection is stopped when this bit is cleared and restarted when this bit is set to 1. The host driver may clear the Card INT STAT Enable before servicing the Card INT and may set this bit again after all INT requests from the card are cleared to prevent inadvertent INTs. 0 - Masked 1 - Enabled
7	CARDRMEN	0	RW	**Card Removal Signal Enable** 0 - Masked 1 - Enabled
6	CARDINSEN	0	RW	**Card Insertion Signal Enable** 0 - Masked 1 - Enabled
5	BUFRDRDYEN	0	RW	**Buffer Read Ready Signal Enable** 0 - Masked 1 - Enabled
4	BUFWRRDYEN	0	RW	**Buffer Write Ready Signal Enable** 0 - Masked 1 - Enabled
3	DMAINTEN	0	RW	**DMA Interrupt Signal Enable** 0 - Masked 1 - Enabled
2	BLKGAPEVTEN	0	RW	**Block Gap Event Signal Enable** 0 - Masked 1 - Enabled
1	TRANCOMEN	0	RW	**Transfer Complete Signal Enable** 0 - Masked 1 - Enabled
0	CMDCOMEN	0	RW	**Command Complete Signal Enable** 0 - Masked 1 - Enabled

39.5.15 SDIO_IEN - Normal and Error Interrupt Signal 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				0		0	0	0	0	0	0	0	0	0	0	0		0	0	0				0	0	0	0	0	0	0	0	0
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				TARGETRESPERRSEN		TUNINGERRSIGNALENABLE	ADMAERRSEN	AUTOCMDERRSEN	CURRENTLIMITERRSEN	DATENDBITERRSEN	DATCRCERRSEN	DATTOUTERRSEN	CMDINDEXERRSEN	CMDENDBITERRSEN	CMDCRCERRSEN	CMDTOUTERRSEN		BOOTTERMINATESEN	BOOTACKRCVSEN	RETUNINGEVTSEN				CARDINTSEN	CARDREMSEN	CARDINSSEN	BUFRDRDYSEN	BUFWRRDYSEN	DMAINTSEN	BLKGAPEVTSEN	TRANCOMSEN	CMDCOMSEN

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	TARGETRESPERRSEN Error Signal Enable	0	RW	**Target Response Error Signal Enable**
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	TUNINGERRSIGNALENABLE 0 - Masked 1 - Enabled	0	RW	**Tuning Error Signal Enable**
25	ADMAERRSEN 0 - Masked 1 - Enabled	0	RW	**ADMA Error Signal Enable**
24	AUTOCMDERRSEN 0 - Masked 1 - Enabled	0	RW	**Auto CMD12 Error Signal Enable**
23	CURRENTLIMITERRSEN 0 - Masked 1 - Enabled	0	RW	**Current Limit Error Signal Enable**
22	DATENDBITERRSEN 0 - Masked 1 - Enabled	0	RW	**Data End Bit Error Signal Enable**
21	DATCRCERRSEN 0 - Masked 1 - Enabled	0	RW	**Data CRC Error Signal Enable**
20	DATTOUTERRSEN 0 - Masked 1 - Enabled	0	RW	**Data Timeout Error Signal Enable**
19	CMDINDEXERRSEN 0 - Masked 1 - Enabled	0	RW	**Command Index Error Signal Enable**

Bit	Name	Reset	Access	Description
18	CMDENDBITERRSEN	0	RW	**Command End Bit Error Signal Enable**
	0 - Masked 1 - Enabled			
17	CMDCRCERRSEN	0	RW	**Command CRC Error Signal Enable**
	0 - Masked 1 - Enabled			
16	CMDTOUTERRSEN	0	RW	**Command Timeout Error Signal Enable**
	0 - Masked 1 - Enabled			
15	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
14	BOOTTERMINATESEN	0	RW	**Boot Terminate Interrupt Signal Enable**
	0 - Masked 1 - Enabled			
13	BOOTACKRCVSEN	0	RW	**Boot Ack Received Signal Enable**
	0 - Masked 1 - Enabled			
12	RETUNINGEVTSEN	0	RW	**Re-Tuning Event Signal Enable**
	0 - Masked 1 - Enabled			
11:9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	CARDINTSEN	0	RW	**Card Interrupt Signal Enable**
	0 - Masked 1 - Enabled			
7	CARDREMSEN	0	RW	**Card Removal Signal Enable**
	0 - Masked 1 - Enabled			
6	CARDINSSEN	0	RW	**Card Insertion Signal Enable**
	0 - Masked 1 - Enabled			
5	BUFRDRDYSEN	0	RW	**Buffer Read Ready Signal Enable**
	0 - Masked 1 - Enabled			
4	BUFWRRDYSEN	0	RW	**Buffer Write Ready Signal Enable**
	0 - Masked 1 - Enabled			
3	DMAINTSEN	0	RW	**DMA Interrupt Signal Enable**
	0 - Masked 1 - Enabled			
2	BLKGAPEVTSEN	0	RW	**Block Gap Event Signal Enable**
	0 - Masked 1 - Enabled			
1	TRANCOMSEN	0	RW	**Transfer Complete Signal Enable**
	0 - Masked 1 - Enabled			
0	CMDCOMSEN	0	RW	**Command Complete Signal Enable**
	0 - Masked 1 - Enabled			

39.5.16 SDIO_AC12ERRSTAT - AUTO CMD12 Error Status and Host Control2 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	0	0							0	0	0x0		0	0x0											0			0	0	0	0	0
Access	RW	RW							RW	RWH	RW		RW	RW											R			R	R	R	R	R
Name	PRSTVALEN	ASYNCINTEN							SAMPCLKSEL	EXETUNING	DRVSTNSEL		SIGEN1P8V	UHSMODESEL											CNIBAC12ERR			AC12INDEXERR	AC12ENDBITERR	AC12CRCERR	AC12TOE	AC12NOTEXE

Bit	Name	Reset	Access	Description
31	PRSTVALEN	0	RW	**Preset Value Enable**
	Host controller Version 3.00 supports this bit. As the operating SD_CLK frequency and I/O driver strength depend on the Host SYS implementation, it is difficult to determine these parameters in the Standard Host Driver. When Preset Value Enable is set to automatic. This bit enables the functions defined in the Preset Value registers. 1 Automatic Selection by Preset Value are Enabled 0 SD_CLK and Driver Strength are controller by Host Driver If this bit is set to 0, SD_CLK Frequency SEL, Clock Generator SEL in the Clock CTRL register and Driver Strength SEL in Host CTRL 2 register are set by Host Driver. If this bit is set to 1, SD_CLK Frequency SEL, Clock Generator SEL in the Clock CTRL register and Driver Strength SEL in Host CTRL 2 register are set by Host controller as specified in the Preset Value registers.			
30	ASYNCINTEN	0	RW	**Asynchronous Interrupt Enable**
	This bit can be set to 1 if a card support asynchronous INT and Asynchronous INT Support is set to 1 in the Capabilities register. Asynchronous INT is effective when DAT[1] INT is used in 4-bit SD mode (and zero is set to INT Pin SEL in the Shared Bus CTRL register). If this bit is set to 1, the Host Driver can stop the SD_CLK during asynchronous INT period to save power. During this period, the Host controller continues to deliver Card INT to the host when it is asserted by the Card. 1 - Enabled, 0- Disabled			
29:24	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
23	SAMPCLKSEL	0	RW	**Sampling Clock Select**
	This bit is set by tuning procedure when Execute Tuning is cleared. Writing 1 to this bit is meaningless and ignored. Setting 1 means that tuning is completed successfully and setting 0 means that tuning is failed. Host controller uses this bit to select sampling clock to receive CMD and DAT. This bit is cleared by writing 0. Change of this bit is not allowed while the Host controller is receiving RESP or a read data block. 1 Tuned clock is used to sample data 0 Fixed clock is used to sample data			
22	EXETUNING	0	RWH	**Execute Tuning**
	This bit is set to 1 to start tuning procedure and automatically cleared when tuning procedure is completed. The result of tuning is indicated to Sampling Clock SEL. Tuning procedure is aborted by writing 0 for more detail about tuning procedure. 1- Execute Tuning, 0 Not Tuned or Tuning Completed			
21:20	DRVSTNSEL	0x0	RW	**Driver Strength Select**
	Host controller output driver in 1.8V signaling is selected by this bit. In 3.3V signaling, this field is not effective. This field can be set depends on Driver Type A, C and D support bits in the Capabilities register. This bit depends on setting of Preset Value Enable. If Preset Value Enable = 0, this field is set by Host Driver. If Preset Value Enable = 1, this field is automatically set by a value specified in the one of Preset Value registers.			

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected

Bit	Name	Reset	Access	Description
	2	TYPEC		Driver Type C is selected
	3	TYPED		Driver Type D is selected
19	SIGEN1P8V	0	RW	**Voltage 1.8V Signal Enable**
	This bit controls voltage regulator for I/O cell. 3.3V is supplied to the card regardless of signaling voltage. Setting this bit from 0 to 1 starts changing signal voltage from 3.3V to 1.8V. 1.8V regulator output shall be stable within 5ms. Host Controller clears this bit if switching to 1.8V signaling fails. Clearing this bit from 1 to 0 starts changing signal voltage from 1.8V to 3.3V. 3.3V regulator output shall be stable within 5ms. Host Driver can set this bit to 1 when Host controller supports 1.8V signaling (One of support bits is set to 1: SDR50, SDR104 or DDR50 in the Capabilities register) and the card or device supports UHS-I 1 - 1.8V Signaling 0 - 3.3V Signaling			
18:16	UHSMODESEL	0x0	RW	**UHS Mode Select**
	This field is used to select one of UHS-I modes and effective when 1.8V Signaling Enable is set to 1. If Preset Value Enable in the Host CTRL 2 register is set to 1, Host controller sets SD_CLK Frequency SEL, Clock Generator SEL in the Clock CTRL register and Driver Strength SEL according to Preset Value registers. In this case, one of preset value registers is selected by this field. Host Driver needs to reset SDCLKEN before changing this field to avoid generating clock glitch. After setting this field, Host Driver sets SDCLKEN again. SDR104 or DDR50 is selected for SDIO card, INT detection at the block gap shall not be used. Read Wait timing is changed for these modes. Refer to the SDIO Specification Version 3.00 for more detail.			
	Value	Mode		Description
	0	SDR12		SDR12
	1	SDR25		SDR25
	2	SDR50		SDR50
	3	SDR104		SDR104
	4	DDR50		DDR50
15:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
7	CNIBAC12ERR	0	R	**Command Not Issued By Auto CMD12 Error**
	Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 ERR (D04 - D01) in this register. This bit is set to 0 when Auto CMD ERR is generated by Auto CMD23 0 - No error 1 - Not Issued			
6:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	AC12INDEXERR	0	R	**Auto CMD Index Error**
	Occurs if the CMD Index ERR occurs in RESP to a command. 0 - No error 1 - error			
3	AC12ENDBITERR	0	R	**Auto CMD End Bit Error**
	Occurs when detecting that the end bit of CMD RESP is 0. 0 - No error 1 - End Bit error Generated			
2	AC12CRCERR	0	R	**Auto CMD CRC Error**
	Occurs when detecting a CRC ERR in the CMD RESP. 0 - No error 1 - CRC error Generated			
1	AC12TOE	0	R	**Auto CMD12 Timeout Error**
	Occurs if the no RESP is returned within 64 SD_CLK cycles from the end bit of the CMD. If this bit is set to 1, the other ERR STAT bits (D04 - D02) are meaningless. 0 - No error 1 - Timeout			
0	AC12NOTEXE	0	R	**Auto CMD12 Not Executed**
	If memory multiple block data transfer is not started due to command error, this bit is not set because it is not necessary to issue Auto CMD12. Setting this bit to 1 means the host controller cannot issue Auto CMD12 to stop memory multiple block transfer due to some error. If this bit is set to 1, other ERR STAT bits (D04 - D01) are meaningless. This bit is set to 0 when Auto CMD ERR is generated by Auto CMD23 0 - Executed 1 - Not Executed			

39.5.17 SDIO_CAPAB0 - Capabilities Register to Hold Bits 31~0

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	0x0		0	0		0	0	0	0	0	0		0	0	0x0		0x00								0		0x00					
Access	R		R	R		R	R	R	R	R	R		R	R	R		R								R		R					
Name	IFSLOTTYPE		ASYNCINTSUP	SYSBUS64BSUP		VOLTSUP1P8V	VOLTSUP3P0V	VOLTSUP3P3V	SUSRESSUP	SDMASUP	HSSUP		ADMA2SUP	EXTMEDIABUSSUP	MAXBLOCKLEN		BASECLKFREQSD								TMOUTCLKUNIT		TMOUTCLKFREQ					

Bit	Name	Reset	Access	Description
31:30	IFSLOTTYPE	0x0	R	**Interface Card Slot Type**

by a specific Host SYS. (A host controller register set is defined per slot. Shared Bus Slot (10b) can be set if Host controller supports Shared Bus CTRL register. The Standard Host Driver controls only a removable card or one embedded device is connected to a SD bus slot. If a slot is configured for shared bus (10b), the Standard Host Driver does not control embedded devices connected to a shared bus. Shared bus slot is controlled by a specific host driver developed by a Host SYS.

Value	Mode	Description
0	REMOVABLE	Removable Card Slot
1	EMBEDDED	Only one non-removable device is conected to a SD bus slot
2	SHARED	Can be set if Host controller supports Shared Bus CTRL register

29	ASYNCINTSUP	0	R	**Asynchronous Interrupt Support**

sion 3.00 about asynchronous INT. 1 Asynchronous INT Supported 0 Asynchronous INT Not Supported

28	SYSBUS64BSUP	0	R	**System Bus 64-bit Support**

0 - Does not support 64 bit SYS ADDR

27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	VOLTSUP1P8V	0	R	**Voltage Support 1.8V**

1 - 1.8 V Supported

25	VOLTSUP3P0V	0	R	**Voltage Support 3.0V**

1 - 3.0 V Supported

24	VOLTSUP3P3V	0	R	**Voltage Support 3.3V**

1 - 3.3 V Supported

23	SUSRESSUP	0	R	**Suspend / Resume Support**

supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism are not supported and the host driver shall not issue either Suspend / Resume CMDs. 0 - Not Supported 1 - Supported

22	SDMASUP	0	R	**SDMA Support**

is capable of using DMA to transfer data between SYS memory and the host controller directly. 0 - SDMA Not Supported 1 - SDMA Supported.

Bit	Name	Reset	Access	Description
21	HSSUP	0	R	**High Speed Support** and the Host SYS support High Speed mode and they can supply SD_CLK frequency from 25-50 MHz (for SD) or 20-52 MHz (for MMC). 0 - High Speed Not Supported 1 - High Speed Supported
20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19	ADMA2SUP	0	R	**ADMA2 Support** 0 - ADMA2 not supported
18	EXTMEDIABUSSUP	0	R	**Extended Media Bus Support** controller is capable of using 8-bit bus width mode. This bit is not effective when Slot Type is set to 10b. In this case, refer to Bus Width Preset in the Shared Bus resister. 1 - Extended Media Bus Supported 0 - Extended Media Bus not Supported
17:16	MAXBLOCKLEN	0x0	R	**Maximum Block Length** block size that the host driver can read and write to the buffer in the host controller. The buffer shall transfer this block size without wait cycles. Three sizes can be defined as indicated below. 00 - 512 byte 01 - 1024 byte 10 - 2048 byte 11 - 4096 bytes
15:8	BASECLKFREQSD	0x00	R	**Base Clock Frequency for SD_CLK** This mode is supported by the Host controller Version 1.00 and 2.00. Upper 2-bit is not effective and always 0. Unit values are 1 MHz. The supported clock range is 10 MHz to 63 MHz. 11xx xxxxb Not supported 0011 1111b 63 MHz 0000 0010b 2 MHz 0000 0001b 1 MHz 0000 0000b Get information via another method (2) 8-bit Base Clock Frequency This mode is supported by the Host controller Version 3.00.Unit values are 1 MHz. The supported clock range is 10 MHz to 255 MHz. FFh 255 MHz 02h 2 MHz 01h 1 MHz 00h Get information via another method If the real frequency is 16.5 MHz, the lager value shall be set 0001 0001b (17 MHz) because the Host Driver use this value to calculate the clock divider value (Refer to the SD_CLK Frequency SEL in the Clock CTRL register.) and it shall not exceed upper limit of the SD_CLK frequency. If these bits are all 0, the Host SYS has to get information via another method.
7	TMOUTCLKUNIT	0	R	**Timeout Clock Unit** This bit shows unit of base clock frequency used to detect Data Timeout ERR. 0 - kHz 1 - MHz
6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	TMOUTCLKFREQ	0x00	R	**Timeout Clock Frequency** base clock frequency used to detect Data Timeout ERR. Not 0 - 1 kHz to 63 kHz or 1 MHz to 63 MHz 000000b - Get Information via another method.

39.5.18 SDIO_CAPAB2 - Capabilities Register to Hold Bits 63~32

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							0	0	0x00								0x0		0		0x0					0	0	0		0	0	0
Access							R	R	R								R		R		R					R	R	R		R	R	R
Name							SPIBLOCKMODE	SPIMODE	CLOCKKMUL								RETUNEMODES		USETUNSDR50		TIMCNTRETUN					DRVTYPDSUP	DRVTYPCSUP	DRVTYPASUP		DDR50SUP	SDR104SUP	SDR50SUP

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	SPIBLOCKMODE	0	R	**SPI Block Mode Support** 0 - Not Supported 1 - Supported
24	SPIMODE	0	R	**SPI Mode Support** 0 - Not Supported 1 - Supported
23:16	CLOCKKMUL	0x00	R	**Clock Multiplier** value of programmable clock generator. Refer to Clock CTRL register. Setting 00h means that Host controller does not support programmable clock generator. FFh Clock Multiplier M = 256 .... 02h Clock Multiplier M = 3 01h Clock Multiplier M = 2 00h Clock Multiplier is Not Supported
15:14	RETUNEMODES	0x0	R	**Re-tuning Modes** capability of a Host controller and how to manage the data transfer length and a Re-Tuning Timer by the Host Driver 00 - Mode1 01 - Mode2 10 - Mode3 There are two re-tuning timings: Re-Tuning Request and expiration of a Re-Tuning Timer. By receiving either timing, the Host Driver executes the re-tuning procedure just before a next CMD issue
13	USETUNSDR50	0	R	**Use Tuning for SDR50** Controller requires tuning to operate in SDR50. (Tuning is always required to operate SDR104.) 1 SDR50 requires tuning 0 SDR50 does not require tuning
12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	TIMCNTRETUN	0x0	R	**Timer Count for Re-Tuning** of the Re-Tuning Timer for Re-Tuning Mode 1 to 3. 0h - Get information via other source 1h = 1 seconds 2h = 2 seconds 3h = 4 seconds 4h = 8 seconds -- n = 2(n-1) seconds -- Bh = 1024 seconds
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6	DRVTYPDSUP	0	R	**Driver Type D Support** Type D for 1.8 Signaling. 1 Driver Type D is Supported 0 Driver Type D is Not Supported
5	DRVTYPCSUP	0	R	**Driver Type C Support** Type C for 1.8 Signaling. 1 Driver Type C is Supported 0 Driver Type C is Not Supported
4	DRVTYPASUP	0	R	**Driver Type a Support** Type A for 1.8 Signaling. 1 Driver Type A is Supported 0 Driver Type A is Not Supported

Bit	Name	Reset	Access	Description
3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	DDR50SUP	0	R	**DDR50 Support** 0 DDR50 is Not Supported
1	SDR104SUP	0	R	**SDR104 Support** 1 SDR104 is Supported 0 SDR104 is Not Supported
0	SDR50SUP	0	R	**SDR50 Support** shall be set to 1. Bit 40 indicates whether SDR50 requires tuning or not. 1 SDR50 is Supported 0 SDR50 is Not Supported

39.5.19 SDIO_MAXCURCAPAB - Maximum Current Capabilities 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												0x00								0x00								0x00				
Access												R								R								R				
Name												MAXCUR1P8VAL								MAXCUR3P0VAL								MAXCUR3P3VAL				

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	MAXCUR1P8VAL	0x00	R	**Maximum Current for 1.8V** Maximum current for 1.8V
15:8	MAXCUR3P0VAL	0x00	R	**Maximum Current for 3.0V** Maximum current for 3.0V
7:0	MAXCUR3P3VAL	0x00	R	**Maximum Current for 3.3V** Maximum current for 3.3V

39.5.20 SDIO_FEVTERRSTAT - Force Event Register for Auto CMD Error Status

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					0	0	0	0	0	0	0	0	0	0	0									0			0	0	0	0	0
Access	R					R	W	W	W	W	W	W	W	W	W	W									W			W	W	W	W	W
Name	VENSPECE					TUNINGE	ADMAE	AC12E	CURLIMITE	DATEBE	DATCRCE	DATTOE	CMDINDXE	CMDEBE	CMDCRCE	CMDTOE									CNIBAC12E			AC12INDXE	AC12EBE	AC12CRCE	AC12TOE	AC12NEX

Bit	Name	Reset	Access	Description
31:28	VENSPECE	0x0	R	**Force Event for Vendox Specific Error Status**
	Vendor Specific Error Status			
27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	TUNINGE	0	R	**Force Event for Tuning Errro**
	Force Event for Target RESP ERR, 1 - INT is generated 0 - No INT			
25	ADMAE	0	W	**Force Event for ADMA Error**
	Force Event for ADMA ERR, 1 - INT is generated 0 - No INT			
24	AC12E	0	W	**Force Event for Auto CMD Error**
	Force Event for Auto CMD ERR, 1 - INT is generated 0 - No INT			
23	CURLIMITE	0	W	**Force Event for Current Limit Error**
	Force Event for Current Limit ERR 1 - INT is generated 0 - No INT			
22	DATEBE	0	W	**Force Event for Data End Bit Error**
	Force Event for Data End Bit ERR 1 - INT is generated 0 - No INT			
21	DATCRCE	0	W	**Force Event for Data CRC Error**
	Force Event for Data CRC ERR 1 - INT is generated 0 - No INT			
20	DATTOE	0	W	**Force Event for Data Timeout Error**
	Force Event for Data Timeout ERR 1 - INT is generated 0 - No INT			
19	CMDINDXE	0	W	**Force Event for Command Index Error**
	Force Event for CMD Index ERR 1 - INT is generated ERR 0 - No INT			
18	CMDEBE	0	W	**Force Event for Command End Bit Error**
	Force Event for CMD End Bit ERR 1 - INT is generated 0 - No INT			
17	CMDCRCE	0	W	**Force Event for Command CRC Error**
	Force Event for CMD CRC ERR 1 - INT is generated ERR 0 - No INT			
16	CMDTOE	0	W	**Force Event for Command Timeout Error**
	Force Event for CMD Timeout error 1 - INT is generated 0 - No INT			
15:8	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
7	CNIBAC12E	0	W	**Force Event for Command Not Issued By Auto CMD12 Error**
	1 - INT is generated 0 - no INT issued by Auto CMD12 error			
6:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	AC12INDXE	0	W	**Force Event for Auto CMD Index Error**
	1 - INT is generated 0 - no INT error			
3	AC12EBE	0	W	**Force Event for Auto CMD End Bit Error**
	1 - INT is generated 0 - no INT error.			
2	AC12CRCE	0	W	**Force Event for Auto CMD CRC Error**
	1 - INT is generated 0 - no INT error.			
1	AC12TOE	0	W	**Force Event for Auto CMD Timeout Error**
	1 - INT is generated 0 - no INT .			
0	AC12NEX	0	W	**Force Event for Command Not Issued By Auto CM12 Not Executed**
	1 - INT is generated 0 - no INT Executed.			

39.5.21 SDIO_ADMAES - ADMA Error 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																														0	0x0	
Access																														R	R	
Name																														ADMALME	ADMAES	

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	ADMALME	0	R	**ADMA Length Mismatch Error**
	This error occurs in the following 2 cases. While Block Count Enable being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length. Total data length can not be divided by the block length. 1 - error 0 - No error			
1:0	ADMAES	0x0	R	**ADMA Error State**
	This field indicates the state of ADMA when ERR is occurred during ADMA data transfer. This field never indicates 10 because ADMA never stops in this state. D01 - D00 : ADMA ERR State when error occurred Contents of SYS_SDR register 00 - ST_STOP (Stop DMA) Points to next of the ERR descriptor 01 - ST_FDS (Fetch Descriptor) Points to the ERR descriptor 10 - Never set this state (Not used) 11 - ST_TFR (Transfer Data) Points to the next of the ERR descriptor			

39.5.22 SDIO_ADSADDR - ADMA System Address 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	0x00000000																															
Access	RW																															
Name	ADSADDR																															

Bit	Name	Reset	Access	Description
31:0	ADSADDR	0x00000000	RW	**ADMA System Address**

This register holds byte ADDR of executing CMD of the Descriptor table. 32-bit ADDR Descriptor uses lower 32- bit of this register. At the start of ADMA, the Host Driver shall set start ADDR of the Descriptor table. The ADMA increments this register ADDR, which points to next line, when every fetching a Descriptor line. When the ADMA ERR INT is generated, this register shall hold valid Descriptor ADDR depending on the ADMA state. The Host Driver shall program Descriptor Table on 32-bit boundary and set 32-bit boundary ADDR to this register. ADMA2 ignores lower 2-bit of this register and assumes it to be 00b. 32-bit ADDR ADMA Register Value 32- bit SYS ADDR xxxxxxxx 00000000h 00000000h xxxxxxxx 00000004h 00000004h .... xxxxxxxx FFFFFFFCh FFFFFFFCh ADDR

39.5.23 SDIO_PRSTVAL0 - Preset Value for Initialization and Default Speed Mode

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					0	0x000										0x0					0	0x000								
Access	R					R	R										R					R	R								
Name	DSPDRVSTVAL					DSPCLKGENVAL	DSPSDCLKFREQVAL										INITDRVSTVAL					INITCLCKGENVAL	INITSDCLKFREQVAL								

Bit	Name	Reset	Access	Description
31:30	DSPDRVSTVAL	0x0	R	**Driver Strength Select Value for Default Speed**

1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling.

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected
2	TYPEC	Driver Type C is selected
3	TYPED	Driver Type D is selected

29:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	DSPCLKGENVAL	0	R	**Clock Generator Select Value for Default Speed**

controller supports programmable clock generator. 1 Programmable Clock Generator 0 Host controller Ver2.00 Compatible Clock Generator

25:16	DSPSDCLKFREQVAL	0x000	R	**SD_CLK Frequency Select Value for Default Speed**

Frequency SEL in the Clock CTRL Register is described by a host SYS.

15:14	INITDRVSTVAL	0x0	R	**Driver Strength Select Value for Initialization**

1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling.

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected
2	TYPEC	Driver Type C is selected
3	TYPED	Driver Type D is selected

13:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
10	INITCLCKGENVAL	0	R	**Clock Generator Select Value for Initialization** Controller supports programmable clock generator. 1 Programmable Clock Generator 0 Host controller Ver2.00 Compatible Clock Generator
9:0	INITSDCLKFREQV-AL	0x000	R	**SD_CLK Frequency Select Value for Initialization** Frequency SEL in the Clock CTRL Register is described by a host SYS.

39.5.24 SDIO_PRSTVAL2 - Preset Value for High Speed and SDR12 Modes

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					0	0x000										0x0					0	0x000									
Access	R					R	R										R					R	R									
Name	SDR12DRVSTVAL					SDR12CLKGENVAL	SDR12SDCLKFREQVAL										HSPDRVSTVAL					HSPCLKGENVAL	HSPSDCLKFREQVAL									

Bit	Name	Reset	Access	Description
31:30	SDR12DRVSTVAL	0x0	R	**Driver Strength Select Value for SDR12**
	1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling.			

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected
2	TYPEC	Driver Type C is selected
3	TYPED	Driver Type D is selected

29:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	SDR12CLKGENVAL	0	R	**Clock Generator Select Value for SDR12**
	controller supports programmable clock generator. 1 Programmable Clock Generator 0 Host controller Ver2.00 Compatible Clock Generator			
25:16	SDR12SDCLKFREQVAL	0x000	R	**SD_CLK Frequency Select Value for SDR12**
	Frequency SEL in the Clock CTRL Register is described by a host SYS.			
15:14	HSPDRVSTVAL	0x0	R	**Driver Strength Select Value for High Speed**
	1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling.			

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected
2	TYPEC	Driver Type C is selected
3	TYPED	Driver Type D is selected

13:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
10	HSPCLKGENVAL	0	R	**Clock Generator Select Value for High Speed** controller supports programmable clock generator. 1 Programmable Clock Generator 0 Host controller Ver2.00 Compatible Clock Generator
9:0	HSPSDCLKFREQV-AL	0x000	R	**SD_CLK Frequency Select Value for High Speed** Frequency SEL in the Clock CTRL Register is described by a host SYS.

39.5.25 SDIO_PRSTVAL4 - Preset Value for SDR25 and SDR50 Modes

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					0	0x000										0x0					0	0x000									
Access	R					R	R										R					R	R									
Name	SDR50DRVSTVAL					SDR50CLCKGENVAL	SDR50SDCLKFREQVAL										SDR25DRVSTVAL					SDR25CLKGENVAL	SDR25SDCLKFREQVAL									

Bit	Name	Reset	Access	Description
31:30	SDR50DRVSTVAL	0x0	R	**Driver Strength Select Value for SDR50**

1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling.

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected
2	TYPEC	Driver Type C is selected
3	TYPED	Driver Type D is selected

29:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	SDR50CLCKGENVAL	0	R	**Clock Generator Select Value for SDR50**

controller supports programmable clock generator. 1 Programmable Clock Generator 0 Host controller Ver2.00 Compatible Clock Generator

25:16	SDR50SDCLKFREQVAL	0x000	R	**SD_CLK Frequency Select Value for SDR50**

Frequency SEL in the Clock CTRL Register is described by a host SYS.

15:14	SDR25DRVSTVAL	0x0	R	**Driver Strength Select Value for SDR25**

1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling.

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected
2	TYPEC	Driver Type C is selected
3	TYPED	Driver Type D is selected

13:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
10	SDR25CLKGENVAL	0	R	**Clock Generator Select Value for SDR25** controller supports programmable clock generator. 1 Programmable Clock Generator 0 Host controller Ver2.00 Compatible Clock Generator
9:0	SDR25SDCLKFREQVAL	0x000	R	**SD_CLK Frequency Select Value for SDR25** Frequency SEL in the Clock CTRL Register is described by a host SYS.

39.5.26 SDIO_PRSTVAL6 - Preset Value for SDR104 and DDR50 Modes

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					0	0x000										0x0					0	0x000								
Access	R					R	R										R					R	R								
Name	DDR50DRVSTVAL					DDR50CLKGENVAL	DDR50SDCLKFREQVAL										SDR104DRVSTVAL					SDR104CLKGENVAL	SDR104SDCLKFREQVAL								

Bit	Name	Reset	Access	Description
31:30	DDR50DRVSTVAL	0x0	R	**Driver Strength Select Value for DDR50**

1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling.

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected
2	TYPEC	Driver Type C is selected
3	TYPED	Driver Type D is selected

29:27	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
26	DDR50CLKGENVAL	0	R	**Clock Generator Select Value for DDR50**

controller supports programmable clock generator. 1 Programmable Clock Generator 0 Host controller Ver2.00 Compatible Clock Generator

25:16	DDR50SDCLKFREQVAL	0x000	R	**SD_CLK Frequency Select Value for DDR50**

Frequency SEL in the Clock CTRL Register is described by a host SYS.

15:14	SDR104DRVSTVAL	0x0	R	**Driver Strength Select Value for SDR104**

1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling.

Value	Mode	Description
0	TYPEB	Driver Type B is selected (Default)
1	TYPEA	Driver Type A is selected
2	TYPEC	Driver Type C is selected
3	TYPED	Driver Type D is selected

13:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
10	SDR104CLKGENV-AL	0	R	**Clock Generator Select Value for SDR104**
	controller supports programmable clock generator. 1 Programmable Clock Generator 0 Host controller Ver2.00 Compatible Clock Generator			
9:0	SDR104SDCLKFREQVAL	0x000	R	**SD_CLK Frequency Select Value for SDR104**
	Frequency SEL in the Clock CTRL Register is described by a host SYS.			

39.5.27 SDIO_BOOTTOCTRL - Boot Timeout Control 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	0x00000000																															
Access	RW																															
Name	BOOTDATTOCNT																															

Bit	Name	Reset	Access	Description
31:0	BOOTDATTOCNT	0x00000000	RW	**Boot Data Timeout Counter Value**
	This value determines the interval by which DAT line time-outs are detected during boot operation for eMMC card. The value is in number of SD_CLK cycles.			

39.5.28 SDIO_SLOTINTSTAT - Slot Interrupt Status 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	0x10								0x02																							0
Access	R								R																							R
Name	VENDVERNUM								SPECVERNUM																							INTSLOT0

Bit	Name	Reset	Access	Description
31:24	VENDVERNUM	0x10	R	**Vendor Version Number**
	The Vendor Version Number is set to 0x10 (1.0)			
23:16	SPECVERNUM	0x02	R	**Host Controller Compliant Spec Version Number**
	The Host controller Version Number is set to 0x02 (SD Host Specification Vesion 3.00).			
15:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	INTSLOT0	0	R	**Interrupt Signal for Slot#0**
	This status bit indicates the OR of INT signal and Wakeup signal for slot			

39.5.29 SDIO_CTRL - Core Control Signals

Offset	Bit Position																													
0x800	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0x0						0x0				0	0	0x00					0
Access															RW						RW				RW	RW	RW					RW
Name															TXDLYMUXSEL						OTAPDLYSEL				OTAPDLYEN	ITAPCHGWIN	ITAPDLYSEL					ITAPDLYEN

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	TXDLYMUXSEL	0x0	RW	**TX Delay Mux Selection** Selects one of four combinations of delay taps on SDIO_DAT output pins. See timing details in the device data sheet for optimal settings.
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	OTAPDLYSEL	0x0	RW	**Selects One of 32 Taps on the SDIO_CLK Pin** This is effective only when OTAPDY_EN is enabled and Tuning is not enabled.
7	OTAPDLYEN	0	RW	**Selective Tap Delay Line Enable on SDIO_CLK Pin** When set to 1 enables selective tap delay line on SD clock pin(SDIO_CLK).
6	ITAPCHGWIN	0	RW	**Gating Signal for Tap Delay Change** Used to gate the output of the Tap delay lines so as to avoid glitches being propagated into the core. This Signal should be assert few clocks before the ITAPDLYSEL changes and should be asserted for few clocks after.
5:1	ITAPDLYSEL	0x00	RW	**Selects One of 32 Taps on the Rxclk_in Line** This is effective only when ITAPDLYEN is enabled and Tuning is not enabled.
0	ITAPDLYEN	0	RW	**Selective Tap Delay Line Enable on Rxclk_in** When set to 1 enables selective tap delay line on Looped back SDIO_CLK pin. This signal along with ITAPDLYSEL selects the amount of delay to be inserted.

39.5.30 SDIO_CFG0 - Core Configuration 0

Offset	Bit Position																															
0x804	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0	0	0	0	0	0	0	0	0x0		0x00								0	0x00						0x00					
Access		RW	RW	RW	RW	RW	RW	RW	RW	RW		RW								RW	RW						RW					
Name		C1P8VSUP	C3P0VSUP	C3P3VSUP	CSUSPRESSUP	CSDMASUP	CHSSUP	CADMA2SUP	C8BITSUP	MAXBLKLEN		BASECLKFREQ								TOUTCLKUNIT	TOUTCLKFREQ						TUNINGCNT					

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	C1P8VSUP	0	RW	**1P8V Support**
	Should be set to 1, only if 1.8V is supported on SD Interface.			
29	C3P0VSUP	0	RW	**3P0V Support**
	Should be set to 1, only if 3.0V is supported on SD Interface.			
28	C3P3VSUP	0	RW	**Core 3P3V Support**
	Suggested value is 1, as 3.3V is the default voltage on the SD Interface.			
27	CSUSPRESSUP	0	RW	**Suspend/Resume Support**
	Suggested value is 1. Can be set to 0, if application doesn't support Suspend/Resume.			
26	CSDMASUP	0	RW	**SDMA Mode Support**
	Suggested value is 1. Can be set to 0, if application doesn't supports SDMA mode.			
25	CHSSUP	0	RW	**High Speed Mode Support**
	Suggested value is 1 (High speed mode is supported by core).			
24	CADMA2SUP	0	RW	**ADMA2 Mode Support**
	Suggested value is 1. Can be set to 0, if application doesn't support ADMA2 mode.			
23	C8BITSUP	0	RW	**8-bit Interface Support**
	Suggested value is 1. Can be set to 0, if application supports only 4-bit SD interface.			
22:21	MAXBLKLEN	0x0	RW	**MAX Block Length of Transfer**
	MAX Block Length supported by Core/device.			

Value	Mode	Description
0	512B	512 Bytes are Selected
1	1024B	1024 Bytes are Selected
2	2048B	2048 Bytes are Selected

20:13	BASECLKFREQ	0x00	RW	**Base Clock Frequency for SD_CLK**
	This is the frequency of the SDIO_CLK pin.			

Bit	Name	Reset	Access	Description
12	TOUTCLKUNIT	0	RW	**Timeout Clock Unit in kHz or MHz**
	Configures Timeout clock Unit in kHz or MHz(0: kHz, 1:MHz).			
11:6	TOUTCLKFREQ	0x00	RW	**Timeout Clock Frequency**
	Configures Timeout clock frequency. Default value is 1			
5:0	TUNINGCNT	0x00	RW	**Tuning Counter Value**
	Configures the number of Taps (phases) of the SDIO_CLK pin.			

39.5.31 SDIO_CFG1 - Core Configuration 1

Offset	Bit Position																														
0x808	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0		0	0x0		0	0x0				0	0	0	0	0	0	0x0		0
Access														RW		RW	RW		RW	RW				RW	RW	RW	RW	RW	RW	RW		RW
Name														ASYNCWKUPEN		SPISUP	RETUNMODES		TUNSDR50	RETUNTMRCTL				CDRVDSUP	CDRVCSUP	CDRVASUP	CDDR50SUP	CSDR104SUP	CSDR50SUP	SLOTTYPE		ASYNCINTRSUP

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	ASYNCWKUPEN	0	RW	**Asynchronous Wakeup Enable**
	asynchronous wakeup enable			
17	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
16	SPISUP	0	RW	**SPI Support**
	SPI Support			
15:14	RETUNMODES	0x0	RW	**Retuning Modes**
	retuning modes			
13	TUNSDR50	0	RW	**Tuning for SDR50**
	tuning for SDR50			
12:9	RETUNTMRCTL	0x0	RW	**Retuning Timer Control**
	retuning timer control			
8	CDRVDSUP	0	RW	**Support Type D Driver**
	This should be set based on whether Driver Type D for 1.8V signaling is supported			
7	CDRVCSUP	0	RW	**Support Type C Driver**
	This should be set based on whether Driver Type C for 1.8V signaling is supported			
6	CDRVASUP	0	RW	**Support Type a Driver**
	This should be set based on whether Driver Type A for 1.8V signaling is supported			
5	CDDR50SUP	0	RW	**Support DDR50**
	When set to 1, core supports DDR50 mode of operation, Can be set to 0 if the application doesn't support DDR50 mode of operation.			
4	CSDR104SUP	0	RW	**Support SDR104**
	When set to 1, core supports SDR104 mode of operation, Can be set to 0 if the application doesn't support SDR104 mode of operation.			
3	CSDR50SUP	0	RW	**Core Support SDR50**
	When set to 1, core supports SDR50 mode of operation, Can be set to 0 if the application doesn't support SDR50 mode of operation.			

Bit	Name	Reset	Access	Description
2:1	SLOTTYPE	0x0	RW	**Slot Type**
	When set to 1, core supports monitoring of asynchronous interrupt.			

Value	Mode	Description
0	RMSDSLOT	Removable SD Card Slot
1	EMSDSLOT	Embedded SD Card Slot
2	SHBUSSLOT	Shared SD Card Slot

0	ASYNCINTRSUP	0	RW	**Asynchronous Interrupt Support**
	When set to 1, core supports monitoring of asynchronous interrupt.			

39.5.32 SDIO_CFGPRESETVAL0 - Core Configuration Preset Value 0

Offset	Bit Position																															
0x80C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0		0	0x000													0x0		0	0x000									
Access				RW		RW	RW													RW		RW	RW									
Name				DSPDRVST		DSPCLKGENEN	DSPSDCLKFREQ													INITDRVST		INITCLKGENEN	INITSDCLKFREQ									

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*		
28:27	DSPDRVST	0x0	RW	**Default Speed Drive Strength**
	Default Speed Drive Strength.			
26	DSPCLKGENEN	0	RW	**Default Speed Clock Gen Enable**
	Default Speed Clock Gen Enable.			
25:16	DSPSDCLKFREQ	0x000	RW	**Preset Value for Default Speed of SD_CLK**
	Preset value for default Speed of SD_CLK.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:11	INITDRVST	0x0	RW	**Initial Drive Strength**
	Initial Drive Strength.			
10	INITCLKGENEN	0	RW	**Initial Clock Gen Enable**
	Initial Clock Gen Enable.			
9:0	INITSDCLKFREQ	0x000	RW	**Initial SD_CLK Frequency**
	Initial SD_CLK Frequency.			

39.5.33 SDIO_CFGPRESETVAL1 - Core Configuration Preset Value 1

Offset	Bit Position																															
0x810	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0		0	0x000													0x0		0	0x000									
Access				RW		RW	RW													RW		RW	RW									
Name				SDR12DRVST		SDR12CLKGENEN	SDR12SDCLKFREQ													HSPDRVST		HSPCLKGENEN	HSPSDCLKFREQ									

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*		
28:27	SDR12DRVST	0x0	RW	**SDR12 Speed Drive Strength**
	SDR12 Speed Drive Strength.			
26	SDR12CLKGENEN	0	RW	**SDR12 Speed Clock Gen Enable**
	SDR12 Speed Clock Gen Enable.			
25:16	SDR12SDCLKFREQ	0x000	RW	**Preset Value for SDR12 Speed of SD_CLK**
	Preset value for SDR12 Speed of SD_CLK.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:11	HSPDRVST	0x0	RW	**High Speed SD Drive Strength**
	High Speed SD Drive Strength.			
10	HSPCLKGENEN	0	RW	**High Speed SD_CLK Gen Enable**
	High Speed SD_CLK Gen Enable.			
9:0	HSPSDCLKFREQ	0x000	RW	**High Speed SD_CLK Frequency**
	High Speed SD_CLK Frequency.			

39.5.34 SDIO_CFGPRESETVAL2 - Core Configuration Preset Value 2

Offset	Bit Position																															
0x814	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0		0	0x000													0x0		0	0x000									
Access				RW		RW	RW													RW		RW	RW									
Name				SDR50DRVST		SDR50CLKGENEN	SDR50SDCLKFREQ													SDR25DRVST		SDR25CLKGENEN	SDR25SDCLKFREQ									

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*		
28:27	SDR50DRVST	0x0	RW	**SDR50 Speed Drive Strength**
	SDR50 Speed Drive Strength.			
26	SDR50CLKGENEN	0	RW	**SDR50 Speed Clock Gen Enable**
	SDR50 Speed Clock Gen Enable.			
25:16	SDR50SDCLKFREQ	0x000	RW	**Preset Value for SDR50 Speed of SD_CLK**
	Preset value for SDR50 Speed of SD_CLK.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:11	SDR25DRVST	0x0	RW	**SDR25 SD Drive Strength**
	SDR25 SD Drive Strength.			
10	SDR25CLKGENEN	0	RW	**SDR25 SD_CLK Gen Enable**
	SDR25 SD_CLK Gen Enable.			
9:0	SDR25SDCLKFREQ	0x000	RW	**SDR25 SD_CLK Frequency**
	SDR25 SD_CLK Frequency.			

39.5.35 SDIO_CFGPRESETVAL3 - Core Configuration Preset Value 3

Offset	Bit Position																															
0x818	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset				0x0		0	0x000													0x0		0	0x000									
Access				RW		RW	RW													RW		RW	RW									
Name				DDR50DRVST		DDR50CLKGENEN	DDR50SDCLKFREQ													SDR104DRVST		SDR104CLKGENEN	SDR104SDCLKFREQ									

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*		
28:27	DDR50DRVST	0x0	RW	**DDR50 Speed Drive Strength**
	DDR50 Speed Drive Strength.			
26	DDR50CLKGENEN	0	RW	**DDR50 Speed Clock Gen Enable**
	DDR50 Speed Clock Gen Enable.			
25:16	DDR50SDCLKFREQ	0x000	RW	**Preset Value for DDR50 Speed of SD_CLK**
	Preset value for DDR50 Speed of SD_CLK.			
15:13	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
12:11	SDR104DRVST	0x0	RW	**SDR104 SD Drive Strength**
	SDR104 SD Drive Strength.			
10	SDR104CLKGENEN	0	RW	**SDR104 SD_CLK Gen Enable**
	SDR104 SD_CLK Gen Enable.			
9:0	SDR104SDCLKFREQ	0x000	RW	**SDR104 SD_CLK Frequency**
	SDR104 SD_CLK Frequency.			

39.5.36 SDIO_ROUTELOC0 - I/O LOCATION Register

Offset	Bit Position																													
0x81C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			CLKLOC								WPLOC								CDLOC								DATLOC					

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	CLKLOC	0x00	RW	**I/O Location for CLK**
	Selects location of CLK pin.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1

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*		
21:16	WPLOC	0x00	RW	**I/O Location for WP**
	Selects location of WP pin.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

Bit	Name	Reset	Access	Description
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	CDLOC	0x00	RW	**I/O Location for CD**
	Selects location of Card detect pin.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

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*		

Bit	Name	Reset	Access	Description
5:0	DATLOC	0x00	RW	**I/O Location for D0-7 Pins**

Selects location of Data pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1

39.5.37 SDIO_ROUTELOC1 - I/O LOCATION Register

Offset	Bit Position																													
0x820	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x00				
Access																											RW				
Name																											CMDLOC				

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	CMDLOC	0x00	RW	**I/O Location for CMD Pin**

Selects location of CMD pin.

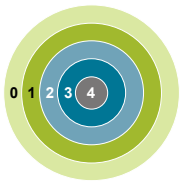
Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1

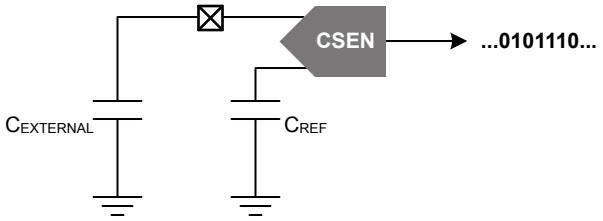
39.5.38 SDIO_ROUTEPEN - I/O LOCATION Enable Register

Offset	Bit Position																													
0x824	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																							0	0	0	0	0	0	0	0	0	0
Access																							RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																							D7PEN	D6PEN	D5PEN	D4PEN	D3PEN	D2PEN	D1PEN	D0PEN	CMDPEN	CLKPEN

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	D7PEN	0	RW	**Data7 I/O Enable**
	When enabled connects output/input of Data7 to pins.			
8	D6PEN	0	RW	**Dat6 Enable**
	When enabled connects output/input of Data6 to pins.			
7	D5PEN	0	RW	**Dat5 Enable**
	When enabled connects output/input of Data5 to pins.			
6	D4PEN	0	RW	**Dat4 I/O Enable**
	When enabled connects output/input of Data4 to pins.			
5	D3PEN	0	RW	**Dat3 I/O Enable**
	When enabled connects output/input of Data3 to pin.			
4	D2PEN	0	RW	**Dat2 I/O Enable**
	When enabled connects output/input of Data2 to pin.			
3	D1PEN	0	RW	**Dat1 I/O Enable**
	When enabled connects output/input of Data1 to pin.			
2	D0PEN	0	RW	**Dat0 I/O Enable**
	When enabled connects output/input of Data0 to pin.			
1	CMDPEN	0	RW	**CMD I/O Enable**
	When enabled connects output/input of CMD to pin.			
0	CLKPEN	0	RW	**CLK I/O Enable**
	When enabled connects output/input of CLK to pin.			

40. CSEN - Capacitive Sense Module





Quick Facts

What?

The capacitive sense (CSEN) module uses a capacitance-to-digital circuit to measure the capacitance of touch-sensitive switches. The module contains an advanced capacitance-to-digital converter that can be configured to take measurements on a single port pin or scan through a group of up to 64 port pins connected via the APORT buses. Port pins/channels can also be shorted together internally to measure the combined capacitance, lowering the required energy usage for wake-on-touch applications. Adjustable maximum capacitance allows for optimal dynamic range, while hardware accumulation and filtering reduce processor computation time. Interrupts can be generated when CSEN completes conversions or when the measured value crosses a threshold.

Why?

The CSEN module is designed to perform extremely low-power autonomous conversions of capacitive touch switches.

How?

The CSEN module uses charge timing techniques to compare external capacitance against internal reference capacitors.

40.1 Introduction

The capacitive sensing (CSEN) module uses a capacitance-to-digital circuit to determine the capacitance on an input pin. The module can take measurements from different physical pins using the APORT multiplexer. In addition, the module can measure multiple pins in sequence using the scan modes, or multiple pins at the same time (bonded together) using the multiple-channel measurement feature. CSEN is available in EM0, EM1, EM2 and EM3.

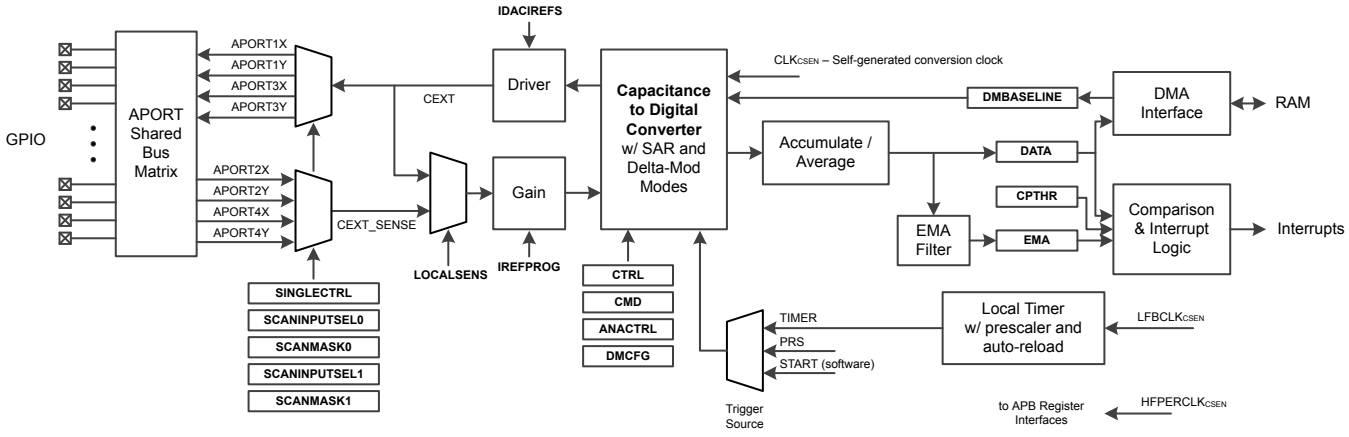


Figure 40.1. CSEN Overview

40.2 Features

- Up to 64 Channels, connected via APORT buses.
- Single sample, continuous single sample, single scan and continuous scan modes supported.
- Support for both SAR and Delta Modulation conversions.
  - SAR conversions are progammable to resolutions of 10, 12, 14, or 16 bits.
  - Delta Modulation conversions support dynamic or fixed gain, and are programmable to resolutions of 10, 12, 14, or 16 bits.
- Supports channel bonding to monitor multiple channels, shorted internally, with a single conversion. The maximum number of channels to be shorted is limited by the maximum capacitance that can be measured by the analog core.
- Conversions triggered by software, PRS, or dedicated timer running from LFXO, LFRCO, or ULFRCO.
- Hardware accumulation of 1, 2, 4, 8, 16, 32, or 64 samples. The result can be right-shifted to the desired resolution or collected in a 22-bit accumulator for further processing by software.
- Hardware exponential moving average (EMA) filter for candidate touch detection during low power autonomous operation.
- Automatic threshold comparison with programmable polarity.
- Low-frequency noise filter to reject noise sources such as 50/60 Hz.
- DMA interface to collect accumulated samples in on-chip RAM. The DMA interface can also be used to program starting values for delta-modulation conversions. The DMA interface can run down to EM2.

40.3 Timing

Conversion timing for the CSEN block is flexible, with access to several clocks and conversion trigger sources.

40.3.1 Clocks

The CSEN module takes two external clock sources as input. The register interface is driven by the HFPERCLK source, allowing for fast access to the control and data for the block. The local timer is driven by the LFBCLK source. The CSEN block also contains an internal, low-energy conversion clock source (CLK_{CSEN}). Conversions are self-timed and the block only requires external clocks under certain circumstances.

For register access, $HFPERCLK_{CSEN}$ must be enabled to the CSEN block in the CMU_HFPERCLKEN0 register. When register access is not required, this clock may be shut down for energy savings.

$LFBCLK_{CSEN}$ is used by the local CSEN timer. It should be enabled to the CSEN block in the CMU_LFBCLKEN0 register any time the CSEN local timer is used as a conversion trigger.

40.3.2 Conversion Triggers

CSEN conversions can be triggered from one of three different sources, selected by CTRL_STM: a software register write, a PRS channel, or the local CSEN timer. The selected trigger source begins a conversion cycle. Depending on the selected conversion mode, one conversion trigger may generate one or many output words from the CSEN module. See 40.6 Converison Modes for more details on the CSEN output for each mode.

Software Triggered Conversions

When CTRL_STM is set to START, conversions are triggered by software. Software triggering is typically used when operating the CSEN block in a continuous mode to gather conversions as quickly as the converter allows. Software triggering may also be used in applications where a single conversion or one scan cycle is needed infrequently and sporadically by different software processes. When configured for software triggering, a write of the CMD_START bit to logic 1 initiates conversions.

PRS Triggered Conversions

When CTRL_STM is set to PRS, conversions are triggered via a PRS event. PRS triggers are typcally used when it is necessary to synchronize CSEN conversions with other events in the system. For example, the LETIMER may be used in EM2 to initiate a CSEN scan operation and other events at the same time. A number of different PRS channels may be configured as the trigger source. The specific PRS channel to be used as a trigger source is selected in the PRSSEL register.

CSEN Timer Triggered Conversions

When CTRL_STM is set to TIMER, a local 8-bit CSEN timer is used to trigger conversions. The local timer is typically used in conjunction with non-continuous conversion modes to provide periodic conversion triggers at slow sampling rates. The local timer is clocked from $LFBCLK_{CSEN}$, and configured with the TIMCTRL register. The CSEN timer has a local prescaler (set by the TIMCTRL_PCPRESC field), which divides the $LFBCLK_{CSEN}$ further. The TIMCTRL_PCTOP field sets the reload value for the CSEN timer. The CSEN timer counts down for the number of clocks specified in TIMCTRL_PCTOP. When the counter reaches zero, a conversion cycle is triggered and the timer is reloaded.

40.3.3 Shutdown and Warmup

Many target applications for CSEN require low power operation and infrequent sampling. By default, the converter will power down when it is not in use to save energy. Upon receiving a conversion trigger, the CSEN block will power on and wait for (3 + TIMCTRL_WARMPCNT) CLK_{CSEN} clock cycles before starting conversions.

CTRL_WARMUPMODE defines the behavior of the CSEN converter when it is not actively converting. Software may choose to keep the CSEN block powered on at all times by setting the CTRL_WARMUPMODE bit to logic 1.

40.4 Conversion Types

The CSEN block offers two different types of conversions: SAR, and Delta Modulation.

40.4.1 SAR Conversion Type

SAR (successive approximation register) conversions are self-contained and do not depend on the results of any previous conversions. SAR conversions will be performed when CTRL_CONVSEL is set to SAR. CTRL_SARCR sets the SAR conversion resolution, and is selectable between 10, 12, 14, or 16 bits. SAR converisons last for N cycles of CLK_{CSEN}, where N is the selected resolution (i.e. 12-bit conversions last for 12 CLK_{CSEN} cycles).

Every SAR conversion consists of a set of tests, one for each bit of the converter. The MSB is tested first, followed by all other bits down to the LSB. Each test narrows the possible value by 1/2 until the final result is determined.

40.4.2 Delta Modulation Conversion Type

Delta modulation (DM) conversions provide significant noise, response time, and energy consumption improvements over SAR conversions. A DM conversion inherently takes longer than a SAR conversion to arrive at one result. However, it is much less susceptible to noise events in the system. Whereas a large number of SAR conversions may need to be averaged to produce a desired noise resolution, the same noise resolution can be achieved with few DM conversions. However, DM conversions require more specific knowledge of the underlying conversion process to implement effectively. Silicon Laboratories software libraries use delta modulation for the best possible performance. It is recommended to use the provided libraries in most applcations. The information provided in this section is intended as a reference for CSEN module driver development.

Delta modulation conversions are performed when CTRL_CONVSEL is set to DM. DMCFG_DMCR sets the DM conversion resolution, and is selectable between 10, 12, 14, or 16 bits. The selected resolution does not have an impact on conversion timing.

A delta modulated conversion begins with a test against a provided initial value, known as the baseline. If the comparison is high (the external capacitor is larger than the initial value), the comparison value is increased by a specified amount (the delta). If the comparison is low, the comparison value is decreased by the delta. The cycle then repeats, testing against the new value. Each subsequent test brings the test value closer to the external capacitance value, until the desired number of tests have been performed and an output result is produced. The converter may be configured to use a fixed delta value, or to reduce the value by a factor of two at specific intervals.

The DM state machine is configured using different fields in the DMCFG register. The DMG field sets the initial delta step to be used, between 0 and 255 codes at the selected resolution (specified by DMCFG_CRMODE). The DMCFG_DMR field specifies how many tests in a row are to be performed using each delta step. The converter will perform (4 x DMCFG_DMR) tests at each delta step for DMCFG_DMR > 0. For DMCFG_DMR = 0, the converter will perform 64 tests at each delta step. The number of tests performed at each delta step is referred to here as a "cycle". The DMCFG_DMCR field specifies how many cycles the state machine will take to produce one conversion output. For DMCFG_DMCR = 0, the number of cycles will be 16. If the DMCFG_DMGRDIS bit is cleared to 0, the delta step will be halved after each cycle. If DMCFG_DMGRDIS is set to 1, all cycles will use a fixed delta step value. Each test performed by the converter requires one CLK_{CSEN}, and so a full DM conversion requires number_of_cycles x tests_per_cycle clocks of CLK_{CSEN}. Figure 40.2 Delta Modulation Conversion With Gain Reduction (DMCFG_DMGDIS = 0) on page 1741 and Figure 40.3 Delta Modulation Conversion With Fixed Delta Step (DMCFG_DMGDIS = 1) on page 1742 show how delta modulation conversions are performed with and without gain reduction.

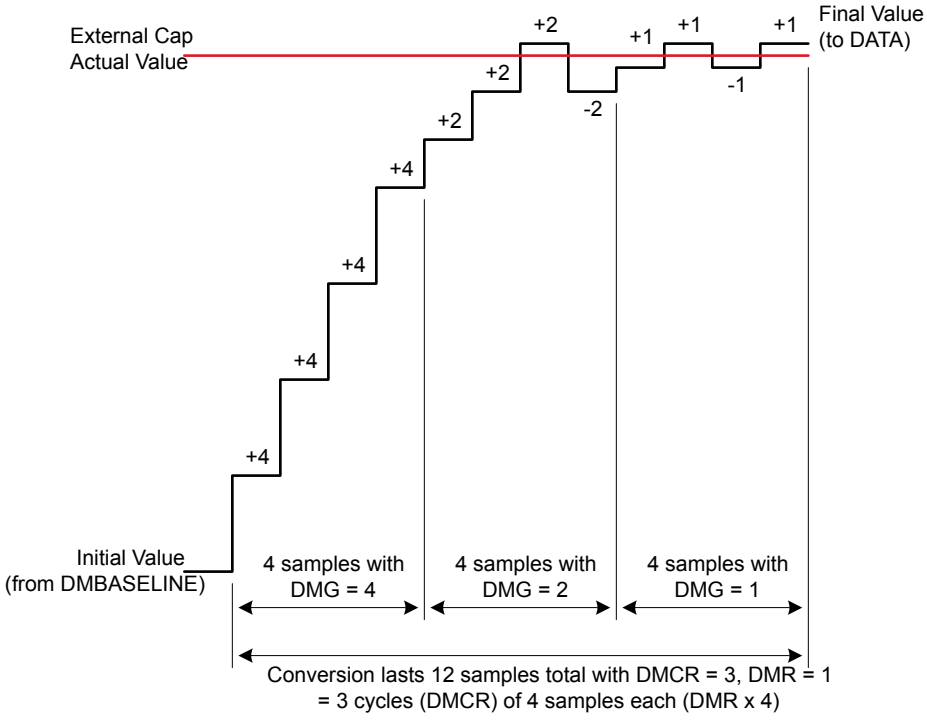


Figure 40.2. Delta Modulation Conversion With Gain Reduction (DMCFG_DMGDIS = 0)

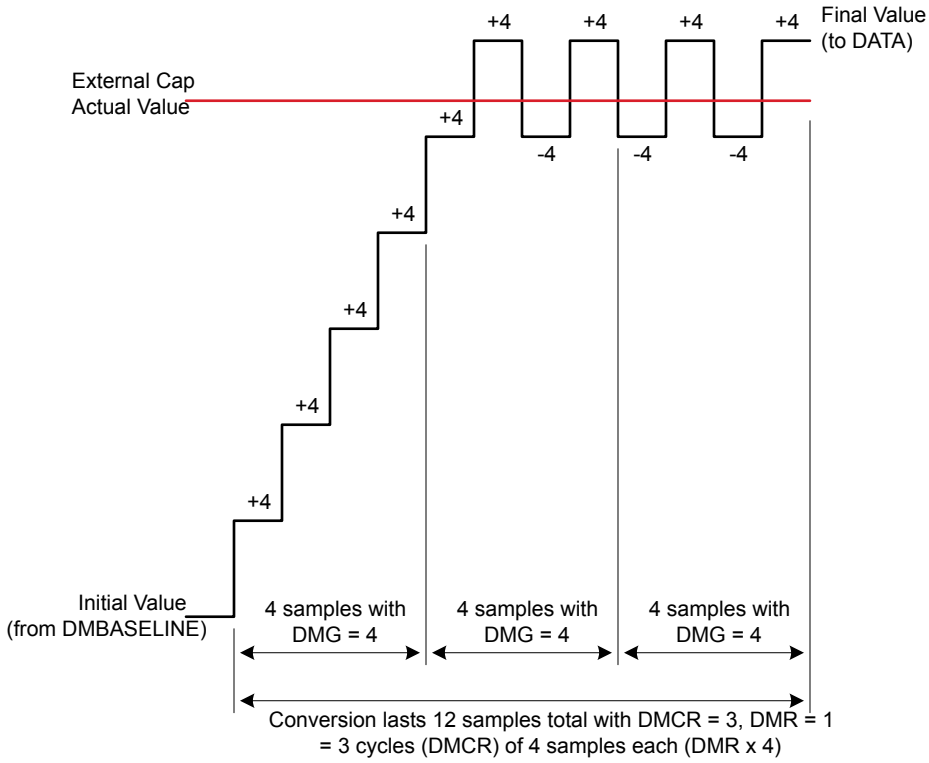


Figure 40.3. Delta Modulation Conversion With Fixed Delta Step (DMCFG_DMGDIS = 1)

The initial test value (baseline) for a DM conversion can be written to the DMBASELINE register prior to the start of the conversion. DMBASELINE contains two fields: BASELINEUP and BASELINEDN. Only the BASELINEUP field is used when chopping is disabled (CTRL_CHOPEN = 0). If chopping is enabled, software must maintain a second baseline value for the ramp-down phase, and write it to BASELINEDN.

40.5 Input Configuration

CSEN channel inputs are routed through the on-chip APORT bus matrix. Each APORT bus is a shared resource among certain analog peripherals on the device. Some knowledge of how the CSEN block utilizes the APORT buses is necessary to avoid conflicts with other peripherals.

Each time the CSEN module connects to an external pin to take a conversion, it uses one or two of the shared APORT buses. By default, the CSEN module charges the external capacitor through one APORT connection on the CEXT line, and senses the voltage at the capacitor through a different APORT connection on the CEXT_SENSE line. This eliminates any errors introduced by series impedance of the APORT buses, particularly if the external cpacitance is large. To conserve on-chip resources, it is possible to both charge and sense through the CEXT line, by setting CTRL_LOCALSENS to logic 1.When CTRL_LOCALSENS is set to 1, the CEXT_SENSE line is not needed and the corresponding bus may be used by other peripherals.

The CSEN module has four APORT buses connected to the CEXT signal and four APORT buses connected to the CEXT_SENSE signal. The specific buses used for each pin selection depend on the channel, according to Table 40.1 CSEN APORT Bus Connectivity on page 1743. For example, if APORT1XCH4 (APORT1 channel 4) is selected in SINGLECTRL_SINGLESEL, the CEXT signal will be connected to the shared BUSAX, channel 4. If CTRL_LOCALSENS is set to DISABLE, then the CEXT_SENSE signal will be connected to APORT2YCH4, and use the shared bus BUSBY, channel 4.

The same connections apply to scan mode conversions and bonded conversions. For scan mode conversions, the connection through the bus is only made on one channel at a time, when that channel is being converted in the scan. For bonded connections, all selected channel connections are made simultaneously.

Table 40.1. CSEN APORT Bus Connectivity

Selected CSEN Channel	CEXT Routing	CEXT_SENSE Routing (LOCALSENS = 0)
APORT1, Even Channel	APORT1XCHn / BUSAXCHn	APORT2YCHn / BUSBYCHn
APORT1, Odd Channel	APORT1YCHn / BUSAYCHn	APORT2XCHn / BUSBXCHn
APORT3, Even Channel	APORT3XCHn / BUSCXCHn	APORT4YCHn / BUSDYCHn
APORT3, Odd Channel	APORT3YCHn / BUSCYCHn	APORT4XCHn / BUSDXCHn

When the CSEN module requests an APORT bus that is already in use by another peripheral, an APORT conflict will occur. The IF_APORTCONFLICT interrupt flag will be set to 1 (generating an interrupt if enabled), and the APORTCONFLICT register will reflect the bus(es) where the conflict occurred. Careful channel planning for the system will avoid APORT conflicts in most systems.

Channel selection for the CSEN module depends on the selected conversion mode. More details on how to select specific input pins are found in 40.6 Converison Modes.

40.6 Converison Modes

The CSEN module supports several conversion modes:

- Single Channel - A conversion trigger starts conversions on a single channel. One output result is produced, then the converter will halt.
- Scan - A conversion trigger starts a scan sequence, which converts a specified number of channels independently and in sequence. One output result is produced per channel, then the converter will halt.
- Bonded Channel - A conversion trigger starts conversions on a bonded channel. A bonded channel is several input channels which are shorted together internally. One output result is produced, then the converter will halt.
- Continuous Single Channel - A conversion trigger starts continuous conversions on a single channel. The single channel conversion will re-trigger automatically after each output result and repeat until halted.
- Continuous Scan - A conversion trigger starts continuous channel scanning. The scan sequence will re-trigger automatically at the end of each scan and repeat until halted.
- Continuous Bonded Channel - A conversion trigger starts continuous conversions on a bonded channel. The bonded channel conversion will re-trigger automatically after each output result and repeat until halted.

The conversion mode is selected by the CTRL_CM and CTRL_MCEN fields. Refer to 40.6.1 Single Channel Conversions, 40.6.2 Scan Conversions, and 40.6.3 Bonded Channel Conversions for specific information on configuring the CSEN module to the desired conversion mode.

40.6.1 Single Channel Conversions

CSEN is configured for single channel mode when CTRL_MCEN = DISABLE and CTRL_CM = SGL. For continuous single channel mode, CTRL_MCEN = DISABLE and CTRL_CM = CONTSGL.

In single channel mode, SINGLECTRL_SINGLESEL specifies the input channel to be converted. Firmware may select any of the pins connected to the CEXT signal via the APORT. Refer to the Analog Port (APORT) Client Maps section in the product data sheet for mapping of the CEXT signal to specific pins.

When a single channel conversion is triggered, the CSEN block will use the configured conversion type (SAR or DM) to convert the capacitance seen at the selected input pin to a digital value one or more times. The hardware accumulator setting (CTRL_ACU) determines how many times the input pin will be sampled and accumulated before an output word is produced. The IF_CONV interrupt flag will be set to 1 by hardware when an output word is available in the DATA register.

Note: The auto-ground feature is typically not used in single channel conversion mode and should be disabled by software. However, if auto-grounding is enabled, it will ground the unused channels specified by the SCANINPUTSEL0/1 and SCANMASKSEL0/1 registers. See the channel selection discussion in 40.6.2 Scan Conversionsfor more details.

40.6.2 Scan Conversions

CSEN is configured for scan mode when CTRL_MCEN = DISABLE and CTRL_CM = SCAN. For continuous scan mode, CTRL_MCEN = DISABLE and CTRL_CM = CONTSCAN.

In scan mode, the input channels to be converted during the scan are determined by the SCANINPUTSEL0/1 and SCANMASK0/1 registers. There are 64 available channels in the scanner logic. SCANINPUTSEL0/1 allow software to route CSEN scan channels to APORT-capable GPIO in groups of 8. SCANMASK0/1 specify which channels on those groups are to be included in the scan.

Note: It is possible to include the same group of 8 in more than one place in the scan sequence. However, for the majority of use cases, the SCANINPUTSEL0 register should be written to 0x07060504, and SCANINPUTSEL1 should be written to 0x0F0E0D0C. This will configure the CSEN scan channels to match their order in bonded mode.

Individual channels are included in the scan based on their bit positions in the mask registers, SCANMASK0 and SCANMASK1. A '1' in the corresponding bit of the mask register will include that channel in the scan. Refer to the Analog Port (APORT) Client Maps section in the product data sheet for mapping of the CEXT signal to specific pins.

When a scan conversion is triggered, the CSEN block will convert each of the selected channels in turn, starting at channel 0 and working up to channel 63. If a channel is not configured for scan in the SCANMASK0/1 register (i.e. the bit for that channel is '0'), it will be skipped.

An example of how the scan logic progresses through channels is shown in Figure 40.4 Scan Mode Sequencing on page 1745. In this simple example, SCANMASK0 = 0x00000318 and SCANMASK1 = 0x00000006, which selects channels 3, 4, 8, 9, 33, and 34 in turn.

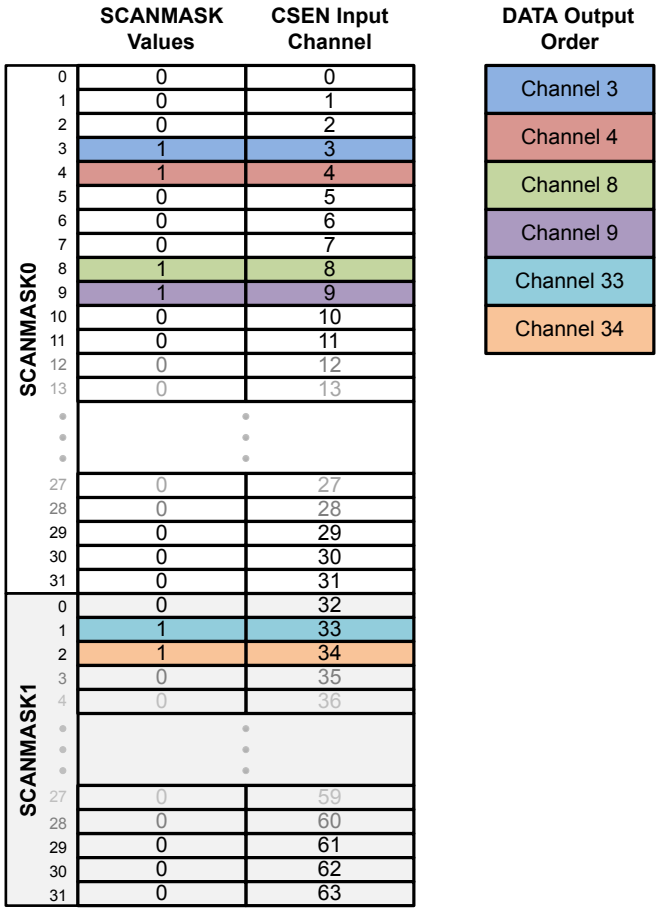


Figure 40.4. Scan Mode Sequencing

An auto-grounding feature is available for scan mode conversions. This feature forces hardware to ground all of the non-active CSEN channels selected by SCANMASK0/1, to reduce bleed-through from measurements on other CSEN channels. Auto-grounding is enabled by setting the CTRL_AUTOGND bit to logic 1.

CSEN will use the configured conversion type (SAR or DM) to convert the capacitance seen at each of the selected input pins to a digital value one or more times. The hardware accumulator setting (CTRL_ACU) determines how many times each input pin will be sampled and accumulated before an output word is produced. The IF_CONV interrupt flag will be set to 1 by hardware when each output word is available in the DATA register, and the IF_EOS interrupt flag will be set to 1 by hardware at the completion of a scan cycle.

The DMBASELINE register provides the starting point for DM conversions. This starting point will be different for each channel in the scan, and the DMBASELINE register should be updated accordingly. This is typically done via DMA.

Note: DMA or software must read the output words from the DATA register before completion of the next conversion. New conversions will over-write the DATA register contents.

40.6.3 Bonded Channel Conversions

CSEN is configured for bonded mode when CTRL_MCEN = ENABLE and CTRL_CM = SGL. For continuous bonded mode, CTRL_MCEN = ENABLE and CTRL_CM = CONTSGL.

Bonded channel conversions are intended primarily for wake-on-touch applications where minimal power consupmtion is necessary. Bonded channel conversions operate similar to single channel conversions, except that multiple channels are shorted together and converted as a single capacitance value.

In bonded mode, the channels to be bonded are based on their bit positions in the mask registers SCANMASK0 and SCANMASK1, according to a fixed location, as shown in Figure 40.5 CSEN Input Configuration in Bonded Mode on page 1746. SCANMASK0 selects channels from APORT1 and SCANMASK1 selects channels from APORT3. Even-numbered channels are connected through the X bus, and odd-numbered channels are connected through the Y bus. Channels with a '1' in the corresponding position of SCANMASK0/1 will be shorted together during the conversion and converterd as a single channel. Refer to the Analog Port (APORT) Client Maps section in the product data sheet for mapping of the CEXT signal to specific pins.

SCANMASK0	CSEN Input Channel	APORT Connection	Shared Bus Channel
0	0	APORT1XCH0	BUSAX channel 0
1	1	APORT1YCH1	BUSAY channel 1
2	2	APORT1XCH2	BUSAX channel 2
3	3	APORT1YCH3	BUSAX channel 3
4	4	APORT1XCH4	BUSAY channel 4
•	•	•	•
•	•	•	•
•	•	•	•
27	27	APORT1YCH27	BUSAY channel 27
28	28	APORT1XCH28	BUSAX channel 28
29	29	APORT1YCH29	BUSAY channel 29
30	30	APORT1XCH30	BUSAX channel 30
31	31	APORT1YCH31	BUSAY channel 31

SCANMASK1	CSEN Input Channel	APORT Connection	Shared Bus Channel
0	32	APORT3XCH0	BUSCX channel 0
1	33	APORT3YCH1	BUSCY channel 1
2	34	APORT3XCH2	BUSCX channel 2
3	35	APORT3YCH3	BUSCX channel 3
4	36	APORT3XCH4	BUSCY channel 4
•	•	○	○
•	•	○	○
•	•	○	○
27	59	APORT3YCH27	BUSCY channel 27
28	60	APORT3XCH28	BUSCX channel 28
29	61	APORT3YCH29	BUSCY channel 29
30	62	APORT3XCH30	BUSCX channel 30
31	63	APORT3YCH31	BUSCY channel 31

Figure 40.5. CSEN Input Configuration in Bonded Mode

When a bonded conversion is triggered, the CSEN block will use the configured conversion type (SAR or DM) to convert the total capacitance seen at the selected input pins to a digital value one or more times. The hardware accumulator setting (CTRL_ACU) determines how many times the input pins will be sampled and accumulated before an output word is produced. The IF_CONV interrupt flag will be set to 1 by hardware when an output word is available in the DATA register.

Note: The auto-ground feature should not be used in bonded conversion mode. Software should clear CTRL_AUTOGND to 0 when configuring CSEN for bonded conversions.

40.7 Output Data

Output data from the CSEN module is posted to the DATA register. The data encoding can be affected by several configuration settings.

Figure 40.6 Data Encoding for Different Resolution Settings on page 1747 shows the effect that the resolution settings (CTRL_SARCR for SAR conversions or DMCFG_CRMODE for DM conversions) have on the output word. In this example, only one sample is accumulated. Regardless of the resolution setting, the MSB is always presented in bit position 15.

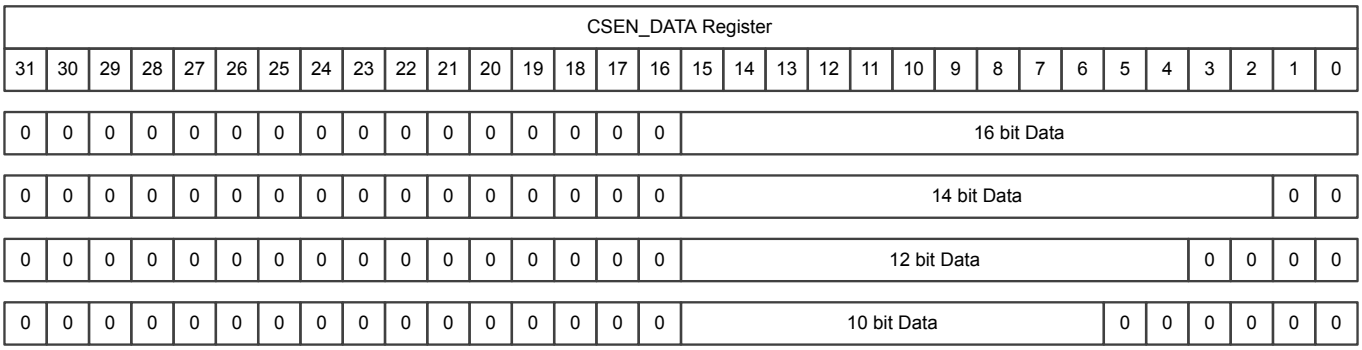


Figure 40.6. Data Encoding for Different Resolution Settings

A hardware accumulator allows the CSEN block to accumulate multiple samples on the same channel for each conversion data word produced, and automatically right-shift to normalize the data to 16 bits. This is effective as a simple noise filter. The CTRL_ACU field sets the number of samples to be accumulated from 1, 2, 4, 8, 16, 32, or 64 samples. The right-shift operation can optionally be disabled by setting CTRL_DRSF to 1. Figure 40.7 Data Encoding for Different Accumulator Settings on page 1747 shows the effects of the accumulator, with and without right-shifting on the output data word. In this example, the sample resolution is fixed at 12 bits, but the same principles apply to any resolution setting.

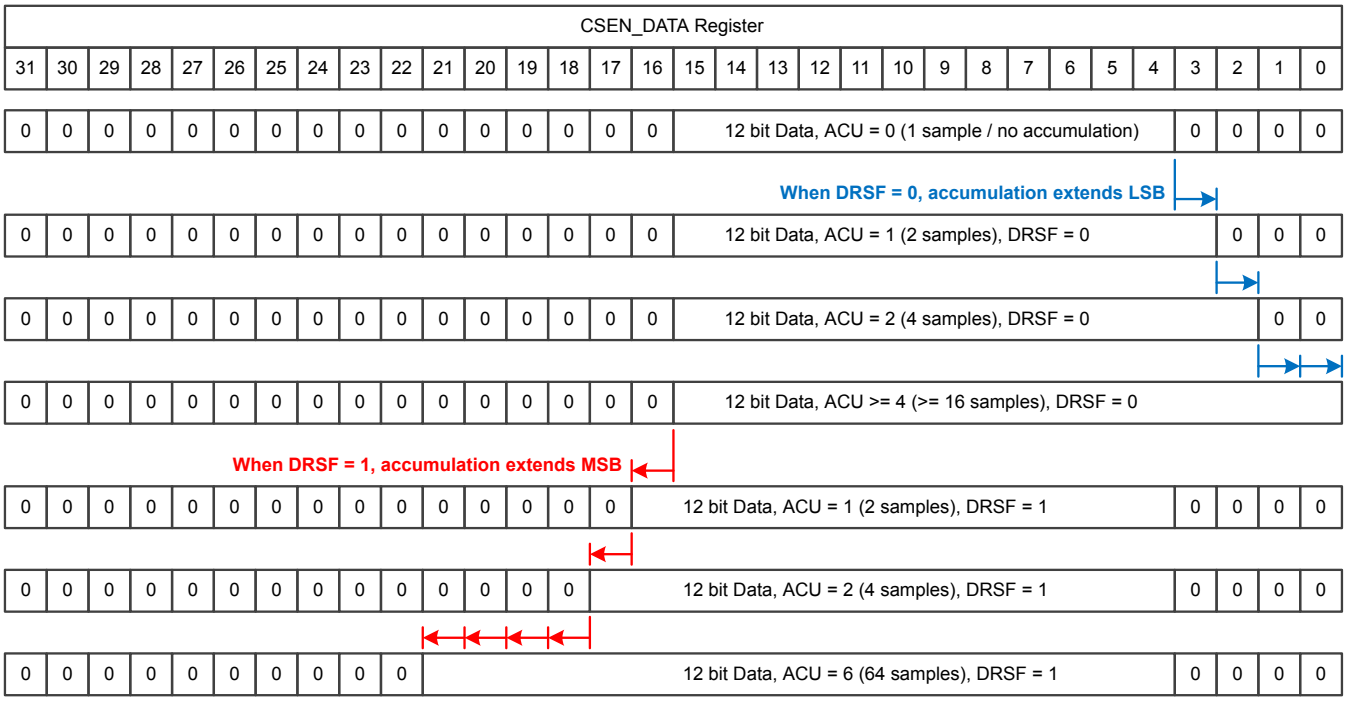


Figure 40.7. Data Encoding for Different Accumulator Settings

When the conversion type is delta modulation (CTRL_CONVSEL = DM) and chopping is enabled (CTRL_CHOPEN = ENABLE), this is a special case for the output word. In this case, both the "up" and the "down" portions of the conversion must be stored. The DATA

register will hold the "up" portion in the lower 16 bits and the "down" portion in the upper 16 bits as shown in Figure 40.8 Data Encoding for Delta Modulation With Chopping on page 1748.

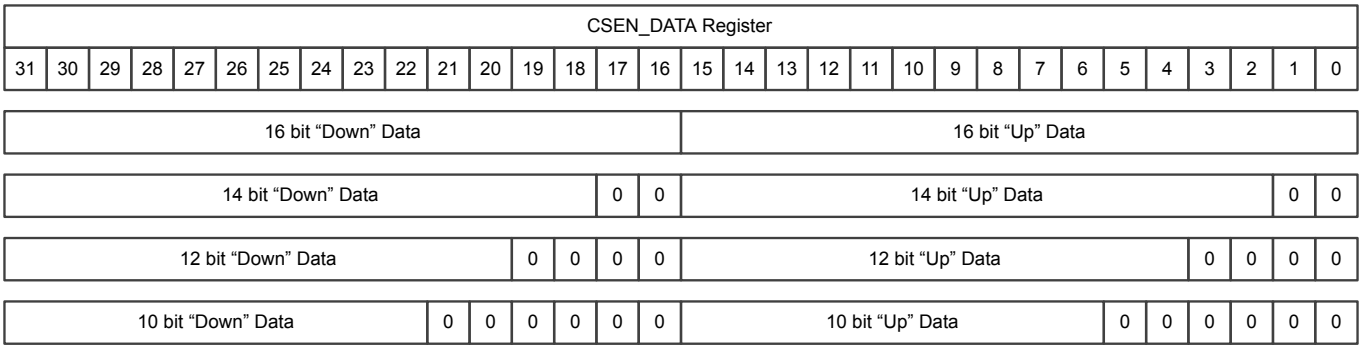


Figure 40.8. Data Encoding for Delta Modulation With Chopping

40.8 Low Frequency Noise Filter (Chopping)

The CSEN module includes a low-frequency noise filter, which is implemented internally using a chopping mechanism. Chopping is enabled by setting CTRL_CHOPEN to ENABLE. In a normal conversion cycle (CTRL_CHOPEN = DISABLE), the charge timing is always performed on a positive (up) ramp. When chopping is enabled, the converter will alternate between using a positive (up) and negative (down) ramp for each sample. While the absolute capacitance value contributes the same amount for an up or a down conversion, any low-frequency offset artifacts due to supply changes will have opposite polarity.Averaging the results of an "up" and a "down" conversion taken back-to-back effectively eliminataes the low-frequency offset differences.

When chopping is used with SAR type converisons, the accumulator performs the necessary averaging in hardware. The accumulator must be set to average at least two samples when chopping is enabled. Because the accumulator always works on multiples of two samples, an even number of "up" and "down" samples will be included in the average.

When chopping is used with DM type conversions, user software must maintain both an "up" and "down" portion of the baseline for the conversion separately. For this reason, the output word will contain both values when using DM conversions. In order to gain the benefits of chopping in DM mode, software should average the "up" and "down" results together.

40.9 Wake on Threshold and Exponential Moving Average

The CSEN module has the capability to operate in the energy-efficient EM2 or EM3 modes and autonomously wake the system when a predetermined threshold is crossed, either while doing single channel conversions or bonded channel conversions. This allows a system to monitor one or more input channels to implement low-energy "wait-for-touch" features. There are two different comparator threshold tests available for this purpose:

- Absolute - used to compare data outputs with "less than or equal" or "greater than" tests against a fixed value. This is useful for applications with short sleep durations, which simply want to wake quickly if a certain condition is met.
- Relative EMA - used to compare data outputs against a moving average window. This is extremely useful for applications with long-term sleep requirements. The relative comparison allows the CSEN module to adjust for slow changes in capacitance such as those due to environmental changes (temperature, supply) without waking the system.

The absolute test is used when CTRL_CMPEN = ENABLE and CTRL_EMACMP = DISABLE. In this mode, any output word written to the DATA register will be compared against the value in the CMPTHR register. The polarity of the comparison is configured with the CTRL_CMPPOL field. Setting CTRL_CMPPOL to GT means that the comparison will be true if DATA is greater than CMPTHR. Setting CTRL_CMPPOL to LTE means that the comparison will be true if DATA is less than or equal to CMPTHR. On a true result, the IF_CMP interrupt flag is set to 1, and any continuous conversion in progress will be halted.

The relative EMA test is used when CTRL_EMACMP = ENABLE. In this mode, the exponential moving average (EMA) value is used to establish a low-noise average code reading. The EMA is a moving average that is re-calculated on every output data word from the converter, according to the following equation:

$$EMA[n] = EMA[n-1] - EMA[n-1]/N + DATA/N,$$

Figure 40.9. CSEN Exponential Moving AverageCalculation

The EMA register stores the current EMA value. This register may be written by software to quickly establish a new baseline average. N in the equation above is the sample weighting of the EMA filter, and is controlled with the EMASAMPLE field. Lower values of EMASAMPLE mean less averaging, and quicker response time, while higher EMASAMPLE values will reject more noise at the expense of response time.

When relative EMA comparisons are enabled, every new sample is compared with a window around the EMA. The lower bound of this window is EMA - CMPTHR, and the upper bound of the window is EMA + CMPTHR. If the new sample written to DATA falls outside that window (lower than the lower bound or higher than the upper bound), the CMP interrupt flag is set to 1 and any continuous conversion in progress is halted. Using the EMA comparison, large jumps in the output data word will trip the comparator and wake the system, but gradual changes will not trigger false positives. Figure 40.10 Wake on Threshold With Relative EMA on page 1750 shows an example of the EMA moving average filter and window.

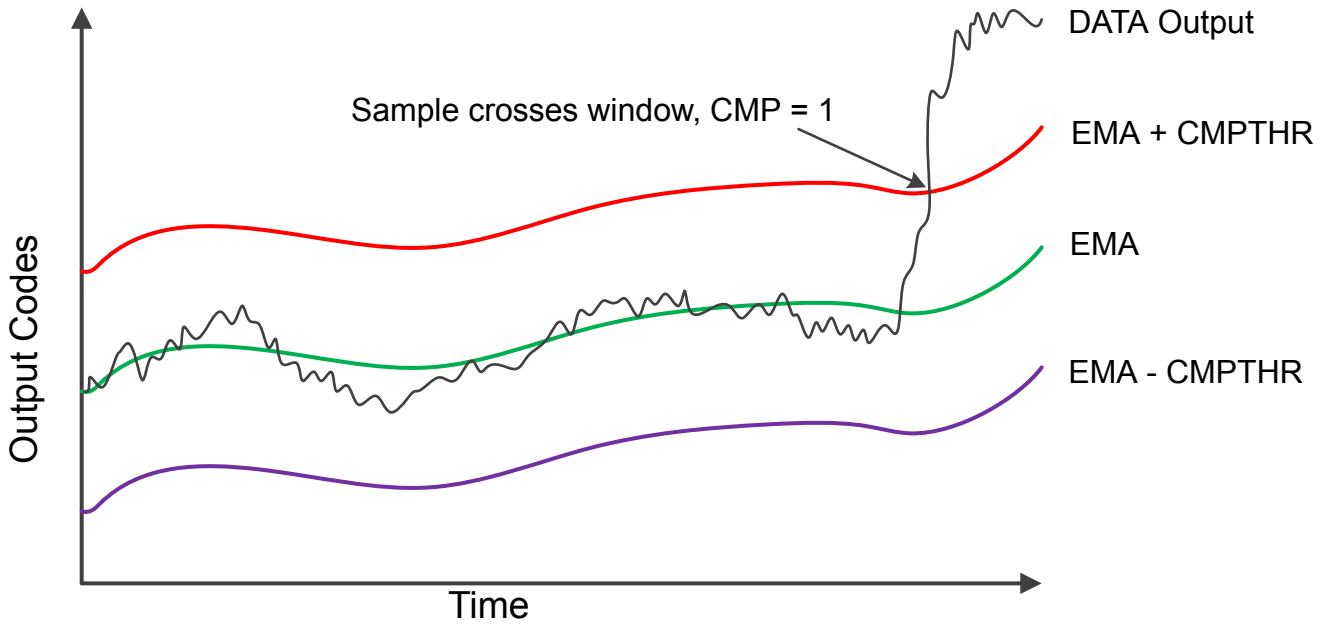


Figure 40.10. Wake on Threshold With Relative EMA

40.10 Analog Adjustments

The analog front-end of the CSEN module has some additional controls that may be useful in certain applications.

40.10.1 Current Reference and Gain

The internal current source used for charging the reference capacitor can be adjusted using the ANACTRL_IREFPROG field. This adjusts the ratio of the internal reference current source vs. the external drive current source, and thereby adjusts the effective gain of the converter. The lowest gain setting is when ANACTRL_IREFPROG = 0, and the highest is with ANACTRL_IREFPROG = 7. The difference between the lowest and highest setting is approximately 10x. High gain gives the best sensitivity and resolution for small capacitors, such as those typically implemented as touch-sensitive PCB features. Lower gain allows for larger capacitor values to be measured. It can also be useful to lower the gain when performing bonded channel conversions.

40.10.2 Current Drive

The external capacitor is charged with a current-source DAC during conversions. The full scale output of the current DAC can be adjusted using the ANACTRL_IDACIREFS field. When ANACTRL_IDACIREFS is 0, the drive current is at its maximum setting. For ANACTRL_IDACIREFS settings of 1-7, the drive current is reduced by a factor of ANACTRL_IDACIREFS / 8. For most touch switch applications, the maximum (default) current drive (ANACTRL_IDACIREFS = 0) should be used. Lower current drive may be useful when there is additional series impedance between the device pin and the capacitive sensor.

40.10.3 Reset (Discharge) Timing

During a conversion, the external capacitance is charged and discharged multiple times. The amount of time used for the reset (discharge) phase is controlled by the ANACTRL_TRSTPROG field. For most touch sensitive switch applications, the fastest (default) timing should be used. Extended reset timing may be useful in applications with additional series impedance between the device pin and the capacitive sensor.

40.11 DMA Interface

The CSEN module has DMA support for reads of the DATA register and writes to the DMBASELINE register. This enables the CSEN module to operate autonomously from software, either to free up software cycles during EM0 or to enable lower power operation in EM1 and EM2.

DMA transfers to and from the CSEN module are enabled by setting the CTRL_DMAEN bit to 1. If the converter is used in SAR mode, only DMA reads (from the DATA register) are triggered. If the converter is used in delta modulation mode, both DMA reads from the DATA register and DMA writes to the DMBASELINE register are triggered.

Requests for a DATA register read occur at the end of a conversion cycle any time the CSEN module writes new output information to the DATA register. DMA may read half words (16 bits) or full words (32 bits) from the DATA register, depending on the specific CSEN configuration and the needs of the application. The CSEN module does not halt when the DMA read request is posted. It will immediately begin the next conversion. If the DMA read request is not serviced by the time the next conversion has completed, the IF_DMAOF flag will be set to indicate an overflow event has occurred.

Requests for a data write to the DMBASELINE register occur only when using delta-modulation type conversions. The write request will be triggered at the start of a conversion, prior to the first sample comparison. The conversion will not begin until the DMA services this write request.

Note: If an absolute or EMA threshold interrupt is enabled when using the DMA, software must be aware that the converter will halt when the interrupt condition occurs. In this case it may be necessary to reconfigure portions of the DMA transfer before resuming conversions.

40.12 Register Map

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

Offset	Name	Type	Description
0x000	CSEN_CTRL	RW	Control
0x004	CSEN_TIMCTRL	RW	Timing Control
0x008	CSEN_CMD	W1	Command
0x00C	CSEN_STATUS	R	Status
0x010	CSEN_PRSSEL	RW	PRS Select
0x014	CSEN_DATA	RWH	Output Data
0x018	CSEN_SCANMASK0	RW	Scan Channel Mask 0
0x01C	CSEN_SCANINPUTSEL0	RW	Scan Input Selection 0
0x020	CSEN_SCANMASK1	RW	Scan Channel Mask 1
0x024	CSEN_SCANINPUTSEL1	RW	Scan Input Selection 1
0x028	CSEN_APORTREQ	R	APORT Request Status
0x02C	CSEN_APORTCONFLICT	R	APORT Request Conflict
0x030	CSEN_CMPTHR	RW	Comparator Threshold
0x034	CSEN_EMA	RWH	Exponential Moving Average
0x038	CSEN_EMACTRL	RW	Exponential Moving Average Control
0x03C	CSEN_SINGLECTRL	RW	Single Conversion Control
0x040	CSEN_DMBASELINE	RW	Delta Modulation Baseline
0x044	CSEN_DMCFG	RW	Delta Modulation Configuration
0x048	CSEN_ANACTRL	RW	Analog Control
0x054	CSEN_IF	R	Interrupt Flag
0x058	CSEN_IFS	W1	Interrupt Flag Set
0x05C	CSEN_IFC	(R)W1	Interrupt Flag Clear
0x060	CSEN_IEN	RW	Interrupt Enable

40.13 Register Description

40.13.1 CSEN_CTRL - 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				0	0	0	0	0	0	0	0	0	0	0	0x3		0	0x0					0x0				0x0			0	0	
Access				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	RW					RW				RW			RW	RW	
Name				CPACCURACY	LOCALSENS	WARMUPMODE	EMACMPEN	MXUC	AUTOGND	CHOPEN	CONVSEL	DMAEN	DRSF	CMPEN	STM		MCEN	ACU					SARCR				CM			CMPPOL	EN	

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	CPACCURACY	0	RW	**Charge Pump Accuracy**

This bit enables a more accurate reference for the APORT supply voltage. For the CSEN module, this bit has no effect on performance and should be written to 0.

Value	Mode	Description
0	LO	Request Low Accuracy Mode.
1	HI	Request High Accuracy Mode.

Bit	Name	Reset	Access	Description
27	LOCALSENS	0	RW	**Local Sensing Enable**

When cleared to 0, Kelvin sensing will be used. The external capacitor will be charged through the CEXT signal and the voltage at the pin will be monitored through a different APORT bus on the CEXT_CSEN signal. When this bit is set to 1, charging and sensing are both performed with the CEXT signal.

Bit	Name	Reset	Access	Description
26	WARMUPMODE	0	RW	**Select Warmup Mode for CSEN**

Use this bit to keep the analog core enabled regardless of the conversion state.

Value	Mode	Description
0	NORMAL	CSEN analog core is shutdown after each operation completes. The next conversion trigger will incur a delay of (3 + WARMUPCNT) CSEN clock cycles before the conversion begins.
1	KEEPCSENWARM	CSEN remains powered up, allowing continuous conversion

Bit	Name	Reset	Access	Description
25	EMACMPEN	0	RW	**Greater and Less Than Comparison Using the Exponential Moving Average (EMA) is Enabled**

The comparator flag (CMPIF) will be set if the accumulated result is greater than or equal to EMA+CMPTHR[8:0] or less than or equal to EMA-CMPTHR[8:0].

Bit	Name	Reset	Access	Description
24	MXUC	0	RW	**CSEN Mux Disconnect**

CSEN Mux Disconnect.

Value	Mode	Description
0	CONN	The CSEN mux inputs are connected.

Bit	Name	Reset	Access	Description
	1	UNC		The CSEN mux inputs unconnected.
23	AUTOGND	0	RW	**CSEN Automatic Ground Enable**

Use this bit to enable automatic grounding of unused channels during scan conversions. For instance, if five channels are included in a scan, four channels will be grounded while one channel is being converted. This feature should be disabled when performing bonded conversions (MCE = 1).

Value	Mode	Description
0	DISABLE	Auto grounding is disabled.
1	ENABLE	Auto grounding is enabled.

22	CHOPEN	0	RW	**CSEN Chop Enable**

Enables chopping for low-frequency noise filtering. When chopping is enabled, the ACU field must be set to a value greater than ACC1.

Value	Mode	Description
0	DISABLE	Chopping is disabled.
1	ENABLE	Chopping is enabled.

21	CONVSEL	0	RW	**CSEN Converter Select**

This bit selects between SAR conversions and delta modulation conversions.

Value	Mode	Description
0	SAR	The CSEN uses the SAR method for conversions.
1	DM	The CSEN uses the delta modulation method for conversions.

20	DMAEN	0	RW	**CSEN DMA Enable Bit**

Enables DMA transfers. DMA triggers are provided for DATA register reads of SAR and DM conversions. DMA triggers are also provided for DMBASELINE writes when DM conversions are performed.

Value	Mode	Description
0	DISABLE	CSEN DMA is disabled.
1	ENABLE	CSEN DMA is enabled.

19	DRSF	0	RW	**CSEN Disable Right-Shift**

Disables the hardware accumulator right-shift operation.

Value	Mode	Description
0	DISABLE	DATA[15:0] stores the last conversion (accumulated) result.
1	ENABLE	DATA[21:0] stores the last conversion (accumulated) result.

18	CMPEN	0	RW	**CSEN Digital Comparator Enable**

This bit enables the digital comparator which compares the accumulated CSEN conversions to the value stored in the CMPTHR register. When enabled, a comparator event will halt continuous conversions. Note that this bit is only effective when EMACMPEN = 0.

Value	Mode	Description
0	DISABLE	CSEN comparator is disabled.

Bit	Name	Reset	Access	Description
	1	ENABLE		CSEN comparator is enabled.
17:16	STM	0x3	RW	**Start Trigger Select**

Selects the start-of-conversion trigger to be used. Depending on the CSEN configuration, a single trigger may result in one or more conversions across multiple channels.

Value	Mode	Description
0	PRS	PRS Triggering. Conversions are triggered by the PRS channel selected in PRSSEL.
1	TIMER	Timer Triggering. Conversions are triggered by a local CSEN timer reload.
2	START	Software Triggering. Conversions are triggered by writing a 1 to the START field of the CMD register.

15	MCEN	0	RW	**CSEN Multiple Channel Enable**

This field enables bonded-channel conversions, where selected channels are shorted together. Use only with CM = SGL or CONTSGL.

Value	Mode	Description
0	DISABLE	Multiple channel feature is disabled.
1	ENABLE	Selected channels are internally shorted together and the combined node is converted.

14:12	ACU	0x0	RW	**CSEN Accumulator Mode Select**

This field configures the hardware accumulator.

Value	Mode	Description
0	ACC1	Accumulate 1 sample.
1	ACC2	Accumulate 2 sample.
2	ACC4	Accumulate 4 sample.
3	ACC8	Accumulate 8 sample.
4	ACC16	Accumulate 16 sample.
5	ACC32	Accumulate 32 sample.
6	ACC64	Accumulate 64 sample.

11:10	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*

9:8	SARCR	0x0	RW	**SAR Conversion Resolution.**

This field selects the resolution of SAR type conversions.

Value	Mode	Description
0	CLK10	Conversions last 10 internal CSEN clocks and are 10-bits in length.
1	CLK12	Conversions last 12 internal CSEN clocks and are 12-bits in length.
2	CLK14	Conversions last 14 internal CSEN clocks and are 14-bits in length.
3	CLK16	Conversions last 16 internal CSEN clocks and are 16-bits in length.

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	CM	0x0	RW	**CSEN Conversion Mode Select**

This field is used to select a conversion method.

Value	Mode	Description
0	SGL	Single Channel Mode: One conversion of a single channel (when MCE = 0) or set of bonded channels (when MCE = 1) per conversion trigger.
1	SCAN	Scan Mode: Scans multiple selected channels once per conversion trigger.
2	CONTSGL	Continuous Single Channel: Continuous conversion of a single channel (when MCE = 0) or set of bonded channels (when MCE = 1).
3	CONTSCAN	Continuous Scan Mode: Continuously scans multiple selected channels.

3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2	CMPPOL	0	RW	**CSEN Digital Comparator Polarity Select**

This bit determines the polarity of the digital comparator.

Value	Mode	Description
0	GT	The digital comparator flag (CMPIF) is set if the conversion result is greater than the threshold.
1	LTE	The digital comparator flag (CMPIF) is set if the conversion result is less than or equal to the threshold.

1	EN	0	RW	**CSEN Enable**

This bit enables the CSEN module. The CSEN module can be configured while disabled and then enabled when conversions are required. Clearing this bit to 0 will cancel any conversions in progress, but does not reset the configuration.

Value	Mode	Description
0	DISABLE	CSEN disabled.
1	ENABLE	CSEN enabled and ready to convert. Must be done before the start trigger.

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

40.13.2 CSEN_TIMCTRL - Timing Control

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		0x00													0x0		
Access															RW		RW													RW		
Name															WARMUPCNT		PCTOP													PCPRESC		

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	WARMUPCNT	0x0	RW	**Warmup Period Counter**
	Configures the warmup time for the converter when WARMUPMODE = NORMAL. The CSEN warmup time is defined as WARMUPCNT+3 CSEN clocks.			
15:8	PCTOP	0x00	RW	**Period Counter Top Value**
	This field contains the reload value for the period counter. If CTRL_STM = TIMER, the counter counts down and a start trigger is generated when the counter reaches zero. The counter is automatically reloaded from this field.			
7:3	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
2:0	PCPRESC	0x0	RW	**Period Counter Prescaler**
	This field sets the pre-scaler for local CSEN timer clock.			

Value	Mode	Description
0	DIV1	The period counter clock frequency is $LFBCLK_{CSEN}$/1
1	DIV2	The period counter clock frequency is $LFBCLK_{CSEN}$/2
2	DIV4	The period counter clock frequency is $LFBCLK_{CSEN}$/4
3	DIV8	The period counter clock frequency is $LFBCLK_{CSEN}$/8
4	DIV16	The period counter clock frequency is $LFBCLK_{CSEN}$/16
5	DIV32	The period counter clock frequency is $LFBCLK_{CSEN}$/32
6	DIV64	The period counter clock frequency is $LFBCLK_{CSEN}$/64
7	DIV128	The period counter clock frequency is $LFBCLK_{CSEN}$/128

40.13.3 CSEN_CMD - Command

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																																0
Access																																W1
Name																																START

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	START	0	W1	**Start Software-Triggered Conversions**
	When CTRL_STM = START, writing a 1 to this bit will trigger CSEN conversions.			

40.13.4 CSEN_STATUS - 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																																0
Access																																R
Name																																CSENBUSY

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	CSENBUSY	0	R	**Busy Flag**
	This bit is set to 1 when a conversion is currently taking place.			

Value	Mode	Description
0	IDLE	Conversion is complete or a conversion is not currently in progress.
1	BUSY	Conversion is in progress.

40.13.5 CSEN_PRSSEL - PRS Select

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																													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	**PRS Channel Select**

Selects the PRS channel to be used as a conversion trigger when CTRL_STM = PRS.

Value	Mode	Description
0	PRSCH0	PRS Channel 0 selected as the start trigger
1	PRSCH1	PRS Channel 1 selected as the start trigger
2	PRSCH2	PRS Channel 2 selected as the start trigger
3	PRSCH3	PRS Channel 3 selected as the start trigger
4	PRSCH4	PRS Channel 4 selected as the start trigger
5	PRSCH5	PRS Channel 5 selected as the start trigger
6	PRSCH6	PRS Channel 6 selected as the start trigger
7	PRSCH7	PRS Channel 7 selected as the start trigger
8	PRSCH8	PRS Channel 8 selected as the start trigger
9	PRSCH9	PRS Channel 9 selected as the start trigger
10	PRSCH10	PRS Channel 10 selected as the start trigger
11	PRSCH11	PRS Channel 11 selected as the start trigger
12	PRSCH12	PRS Channel 12 selected as the start trigger
13	PRSCH13	PRS Channel 13 selected as the start trigger
14	PRSCH14	PRS Channel 14 selected as the start trigger
15	PRSCH15	PRS Channel 15 selected as the start trigger

40.13.6 CSEN_DATA - Output Data

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	0x00000000																															
Access	RWH																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	RWH	**Output Data**
	Output data words are written to the DATA register when sampling and accumulation for a channel have completed. Data encoding depends on the resolution, accumulator settings, and conversion type. See the chapter text for more details on data output encoding.			

40.13.7 CSEN_SCANMASK0 - Scan Channel Mask 0

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	0x00000000																															
Access	RW																															
Name	SCANINPUTEN																															

Bit	Name	Reset	Access	Description
31:0	SCANINPUTEN	0x00000000	RW	**Scan Channel Mask**
	Scan channel mask for CSEN channels CSEN_INPUT0 through CSEN_INPUT31. For scan mode conversions, a '1' in any bit position will include the channel specified by SCANINPUTSEL0 in a scan. For bonded channel conversions, a '1' in any bit position includes the corresponding channel from APORT1 / BUSA in the bonded conversion. If AUTOGND = ENABLE the scan mask also determines the pins that will be grounded when inactive, for both scan and single channel conversions.			

40.13.8 CSEN_SCANINPUTSEL0 - Scan Input Selection 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								0x0								0x0								0x0			
Access					RW								RW								RW								RW			
Name					INPUT24TO31SEL								INPUT16TO23SEL								INPUT8TO15SEL								INPUT0TO7SEL			

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	INPUT24TO31SEL	0x0	RW	**CSEN_INPUT24-31 Select** Channels chosen for CSEN_INPUT24-CSEN_INPUT31 as referred in SCANMASK0

Mode	Value	Description
APORT1CH0TO7	4	Select APORT1 CH0-CH7 as CSEN_INPUT24-CSEN_INPUT31
APORT1CH8TO15	5	Select APORT1 CH8-CH15 as CSEN_INPUT24-CSEN_INPUT31
APORT1CH16TO23	6	Select APORT1 CH16-CH23 as CSEN_INPUT24-CSEN_INPUT31
APORT1CH24TO31	7	Select APORT1 CH24-CH31 as CSEN_INPUT24-CSEN_INPUT31
APORT3CH0TO7	12	Select APORT3 CH0-CH7 as CSEN_INPUT24-CSEN_INPUT31
APORT3CH8TO15	13	Select APORT3 CH8-CH15 as CSEN_INPUT24-CSEN_INPUT31
APORT3CH16TO23	14	Select APORT3 CH16-CH23 as CSEN_INPUT24-CSEN_INPUT31
APORT3CH24TO31	15	Select APORT3 CH24-CH31 as CSEN_INPUT24-CSEN_INPUT31

23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	INPUT16TO23SEL	0x0	RW	**CSEN_INPUT16-23 Select** Channels chosen for CSEN_INPUT16-CSEN_INPUT23 as referred in SCANMASK0

Mode	Value	Description
APORT1CH0TO7	4	Select APORT1 CH0-CH7 as CSEN_INPUT16-CSEN_INPUT23
APORT1CH8TO15	5	Select APORT1 CH8-CH15 as CSEN_INPUT16-CSEN_INPUT23
APORT1CH16TO23	6	Select APORT1 CH16-CH23 as CSEN_INPUT16-CSEN_INPUT23
APORT1CH24TO31	7	Select APORT1 CH24-CH31 as CSEN_INPUT16-CSEN_INPUT23
APORT3CH0TO7	12	Select APORT3 CH0-CH7 as CSEN_INPUT16-CSEN_INPUT23
APORT3CH8TO15	13	Select APORT3 CH8-CH15 as CSEN_INPUT16-CSEN_INPUT23
APORT3CH16TO23	14	Select APORT3 CH16-CH23 as CSEN_INPUT16-CSEN_INPUT23
APORT3CH24TO31	15	Select APORT3 CH24-CH31 as CSEN_INPUT16-CSEN_INPUT23

Bit	Name	Reset	Access	Description
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	INPUT8TO15SEL	0x0	RW	**CSEN_INPUT8-15 Select** Channels chosen for CSEN_INPUT8-CSEN_INPUT15 as referred in SCANMASK0

Mode	Value	Description
APORT1CH0TO7	4	Select APORT1 CH0-CH7 as CSEN_INPUT8-CSEN_INPUT15
APORT1CH8TO15	5	Select APORT1 CH8-CH15 as CSEN_INPUT8-CSEN_INPUT15
APORT1CH16TO23	6	Select APORT1 CH16-CH23 as CSEN_INPUT8-CSEN_INPUT15
APORT1CH24TO31	7	Select APORT1 CH24-CH31 as CSEN_INPUT8-CSEN_INPUT15
APORT3CH0TO7	12	Select APORT3 CH0-CH7 as CSEN_INPUT8-CSEN_INPUT15
APORT3CH8TO15	13	Select APORT3 CH8-CH15 as CSEN_INPUT8-CSEN_INPUT15
APORT3CH16TO23	14	Select APORT3 CH16-CH23 as CSEN_INPUT8-CSEN_INPUT15
APORT3CH24TO31	15	Select APORT3 CH24-CH31 as CSEN_INPUT8-CSEN_INPUT15

7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	INPUT0TO7SEL	0x0	RW	**CSEN_INPUT0-7 Select** Channels chosen for CSEN_INPUT7-CSEN_INPUT0 as referred in SCANMASK0

Mode	Value	Description
APORT1CH0TO7	4	Select APORT1 CH0-CH7 as CSEN_INPUT0-CSEN_INPUT7
APORT1CH8TO15	5	Select APORT1 CH8-CH15 as CSEN_INPUT0-CSEN_INPUT7
APORT1CH16TO23	6	Select APORT1 CH16-CH23 as CSEN_INPUT0-CSEN_INPUT7
APORT1CH24TO31	7	Select APORT1 CH24-CH31 as CSEN_INPUT0-CSEN_INPUT7
APORT3CH0TO7	12	Select APORT3 CH0-CH7 as CSEN_INPUT0-CSEN_INPUT7
APORT3CH8TO15	13	Select APORT3 CH8-CH15 as CSEN_INPUT0-CSEN_INPUT7
APORT3CH16TO23	14	Select APORT3 CH16-CH23 as CSEN_INPUT0-CSEN_INPUT7
APORT3CH24TO31	15	Select APORT3 CH24-CH31 as CSEN_INPUT0-CSEN_INPUT7

40.13.9 CSEN_SCANMASK1 - Scan Channel Mask 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	0x00000000																															
Access	RW																															
Name	SCANINPUTEN																															

Bit	Name	Reset	Access	Description
31:0	SCANINPUTEN	0x00000000	RW	**Scan Channel Mask.**
				Scan channel mask for CSEN channels CSEN_INPUT32 through CSEN_INPUT63. For scan mode conversions, a '1' in any bit position will include the channel specified by SCANINPUTSEL1 in a scan. For bonded channel conversions, a '1' in any bit position includes the corresponding channel from APORT3 / BUSC in the bonded conversion. If AUTOGND = ENABLE the scan mask also determines the pins that will be grounded when inactive, for both scan and single channel conversions.

40.13.10 CSEN_SCANINPUTSEL1 - Scan Input Selection 1

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					INPUT56TO63SEL								INPUT48TO55SEL								INPUT40TO47SEL								INPUT32TO39SEL			

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	INPUT56TO63SEL	0x0	RW	**CSEN_INPUT56-63 Select**

Channels chosen for CSEN_INPUT56-CSEN_INPUT63 as referred in SCANMASK1

Mode	Value	Description
APORT1CH0TO7	4	Select APORT1 CH0-CH7 as CSEN_INPUT56-CSEN_INPUT63
APORT1CH8TO15	5	Select APORT1 CH8-CH15 as CSEN_INPUT56-CSEN_INPUT63
APORT1CH16TO23	6	Select APORT1 CH16-CH23 as CSEN_INPUT56-CSEN_INPUT63
APORT1CH24TO31	7	Select APORT1 CH24-CH31 as CSEN_INPUT56-CSEN_INPUT63
APORT3CH0TO7	12	Select APORT3 CH0-CH7 as CSEN_INPUT56-CSEN_INPUT63
APORT3CH8TO15	13	Select APORT3 CH8-CH15 as CSEN_INPUT56-CSEN_INPUT63
APORT3CH16TO23	14	Select APORT3 CH16-CH23 as CSEN_INPUT56-CSEN_INPUT63
APORT3CH24TO31	15	Select APORT3 CH24-CH31 as CSEN_INPUT56-CSEN_INPUT63

23:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:16	INPUT48TO55SEL	0x0	RW	**CSEN_INPUT48-55 Select**

Channels chosen for CSEN_INPUT48-CSEN_INPUT55 as referred in SCANMASK1

Mode	Value	Description
APORT1CH0TO7	4	Select APORT1 CH0-CH7 as CSEN_INPUT48-CSEN_INPUT55
APORT1CH8TO15	5	Select APORT1 CH8-CH15 as CSEN_INPUT48-CSEN_INPUT55
APORT1CH16TO23	6	Select APORT1 CH16-CH23 as CSEN_INPUT48-CSEN_INPUT55
APORT1CH24TO31	7	Select APORT1 CH24-CH31 as CSEN_INPUT48-CSEN_INPUT55
APORT3CH0TO7	12	Select APORT3 CH0-CH7 as CSEN_INPUT48-CSEN_INPUT55
APORT3CH8TO15	13	Select APORT3 CH8-CH15 as CSEN_INPUT48-CSEN_INPUT55
APORT3CH16TO23	14	Select APORT3 CH16-CH23 as CSEN_INPUT48-CSEN_INPUT55
APORT3CH24TO31	15	Select APORT3 CH24-CH31 as CSEN_INPUT48-CSEN_INPUT55

Bit	Name	Reset	Access	Description
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	INPUT40TO47SEL	0x0	RW	**CSEN_INPUT40-47 Select** Channels chosen for CSEN_INPUT40-CSEN_INPUT47 as referred in SCANMASK1

Mode	Value	Description
APORT1CH0TO7	4	Select APORT1 CH0-CH7 as CSEN_INPUT40-CSEN_INPUT47
APORT1CH8TO15	5	Select APORT1 CH8-CH15 as CSEN_INPUT40-CSEN_INPUT47
APORT1CH16TO23	6	Select APORT1 CH16-CH23 as CSEN_INPUT40-CSEN_INPUT47
APORT1CH24TO31	7	Select APORT1 CH24-CH31 as CSEN_INPUT40-CSEN_INPUT47
APORT3CH0TO7	12	Select APORT3 CH0-CH7 as CSEN_INPUT40-CSEN_INPUT47
APORT3CH8TO15	13	Select APORT3 CH8-CH15 as CSEN_INPUT40-CSEN_INPUT47
APORT3CH16TO23	14	Select APORT3 CH16-CH23 as CSEN_INPUT40-CSEN_INPUT47
APORT3CH24TO31	15	Select APORT3 CH24-CH31 as CSEN_INPUT40-CSEN_INPUT47

7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	INPUT32TO39SEL	0x0	RW	**CSEN_INPUT32-39 Select** Channels chosen for CSEN_INPUT32-CSEN_INPUT39 as referred in SCANMASK1

Mode	Value	Description
APORT1CH0TO7	4	Select APORT1 CH0-CH7 as CSEN_INPUT32-CSEN_INPUT39
APORT1CH8TO15	5	Select APORT1 CH8-CH15 as CSEN_INPUT32-CSEN_INPUT39
APORT1CH16TO23	6	Select APORT1 CH16-CH23 as CSEN_INPUT32-CSEN_INPUT39
APORT1CH24TO31	7	Select APORT1 CH24-CH31 as CSEN_INPUT32-CSEN_INPUT39
APORT3CH0TO7	12	Select APORT3 CH0-CH7 as CSEN_INPUT32-CSEN_INPUT39
APORT3CH8TO15	13	Select APORT3 CH8-CH15 as CSEN_INPUT32-CSEN_INPUT39
APORT3CH16TO23	14	Select APORT3 CH16-CH23 as CSEN_INPUT32-CSEN_INPUT39
APORT3CH24TO31	15	Select APORT3 CH24-CH31 as CSEN_INPUT32-CSEN_INPUT39

40.13.11 CSEN_APORTREQ - APORT Request Status

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																							0	0	0	0	0	0	0	0		
Access																							R	R	R	R	R	R	R	R		
Name																							APORT4YREQ	APORT4XREQ	APORT3YREQ	APORT3XREQ	APORT2YREQ	APORT2XREQ	APORT1YREQ	APORT1XREQ		

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	APORT4YREQ	0	R	**1 If the Bus Connected to APORT4Y is Requested**
	Reports if the bus connected to APORT4Y is being requested from the APORT			
8	APORT4XREQ	0	R	**1 If the Bus Connected to APORT4X is Requested**
	Reports if the bus connected to APORT4X is being requested from the APORT			
7	APORT3YREQ	0	R	**1 If the Bus Connected to APORT3Y is Requested**
	Reports if the bus connected to APORT3Y is being requested from the APORT			
6	APORT3XREQ	0	R	**1 If the Bus Connected to APORT3X is Requested**
	Reports if the bus connected to APORT3X is being requested from the APORT			
5	APORT2YREQ	0	R	**1 If the Bus Connected to APORT2Y is Requested**
	Reports if the bus connected to APORT2Y is being requested from the APORT			
4	APORT2XREQ	0	R	**1 If the Bus Connected to APORT2X is Requested**
	Reports if the bus connected to APORT2X is being requested from the APORT			
3	APORT1YREQ	0	R	**1 If the Bus Connected to APORT1X is Requested**
	Reports if the bus connected to APORT1X is being requested from the APORT			
2	APORT1XREQ	0	R	**1 If the Bus Connected to APORT2X is Requested**
	Reports if the bus connected to APORT1X is being requested from the APORT			
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

40.13.12 CSEN_APORTCONFLICT - APORT Request Conflict

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																							0	0	0	0	0	0	0	0		
Access																							R	R	R	R	R	R	R	R		
Name																							APORT4YCONFLICT	APORT4XCONFLICT	APORT3YCONFLICT	APORT3XCONFLICT	APORT2YCONFLICT	APORT2XCONFLICT	APORT1YCONFLICT	APORT1XCONFLICT		

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	APORT4YCONFLICT	0	R	**1 If the Bus Connected to APORT4Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT4Y is is also being requested by another peripheral
8	APORT4XCONFLICT	0	R	**1 If the Bus Connected to APORT4X is in Conflict With Another Peripheral** Reports if the bus connected to APORT4X is is also being requested by another peripheral
7	APORT3YCONFLICT	0	R	**1 If the Bus Connected to APORT3Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT3Y is is also being requested by another peripheral
6	APORT3XCONFLICT	0	R	**1 If the Bus Connected to APORT3X is in Conflict With Another Peripheral** Reports if the bus connected to APORT3X is is also being requested by another peripheral
5	APORT2YCONFLICT	0	R	**1 If the Bus Connected to APORT2Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT2Y is is also being requested by another peripheral
4	APORT2XCONFLICT	0	R	**1 If the Bus Connected to APORT2X is in Conflict With Another Peripheral** Reports if the bus connected to APORT2X is is also being requested by another peripheral
3	APORT1YCONFLICT	0	R	**1 If the Bus Connected to APORT1Y is in Conflict With Another Peripheral** Reports if the bus connected to APORT1Y is is also being requested by another peripheral
2	APORT1XCONFLICT	0	R	**1 If the Bus Connected to APORT1X is in Conflict With Another Peripheral** Reports if the bus connected to APORT1X is is also being requested by another peripheral
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

40.13.13 CSEN_CMPTHR - Comparator Threshold

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																	0x0000															
Access																	RW															
Name																	CMPTHR															

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	CMPTHR	0x0000	RW	**Comparator Threshold.** When CMPEN is set to 1 and EMACMPEN is cleared to 0, a greater than or less than/equal (based on CMPPOL) comparison between the DATA register and CMPTHR value. If the desired condition is met, the CMPIF flag will be set. When EMACMPEN is set to 1, a comparison window is used instead. The DATA register will be compared against EMA +/- CMPTHR. The CMPIF flag is set any time the conversion result is above (EMA + CMPTHR) or below (EMA - CMPTHR).

40.13.14 CSEN_EMA - Exponential Moving Average

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											0x000000																					
Access											RWH																					
Name											EMA																					

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*		
21:0	EMA	0x000000	RWH	**Calculated Exponential Moving Average** This register contains the current exponential moving average. The EMA is updated every time an accumulated sample is produced, according to the formula: EMA[n] = EMA[n-1] - EMA[n-1]/N + DATA/N, where N is the EMA sample weight selected by EMASAMPLE. This register can be written to initialize the average.

40.13.15 CSEN_EMACTRL - Exponential Moving Average Control

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																															EMASAMPLE	

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	EMASAMPLE	0x0	RW	**EMA Sample Weight**

This field specifies the sample weighting (N) for the exponential moving average filter.

Value	Mode	Description
0	W1	EMA weight (N) is 1.
1	W2	EMA weight (N) is 2.
2	W4	EMA weight (N) is 4.
3	W8	EMA weight (N) is 8.
4	W16	EMA weight (N) is 16.
5	W32	EMA weight (N) is 32.
6	W64	EMA weight (N) is 64.

40.13.16 CSEN_SINGLECTRL - Single Conversion Control

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																						0x00										
Access																						RW										
Name																						SINGLESEL										

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*		
10:4	SINGLESEL	0x00	RW	**Single Channel Input Select**

This field selects the channel to be sampled for single channel conversions.

Mode	Value	Description
APORT1XCH0	32	Select APORT1XCH0
APORT1YCH1	33	Select APORT1YCH1
...	...	.....
APORT3XCH0	96	Select APORT3XCH0
APORT3YCH1	97	Select APORT3YCH1
...	...	.....

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*		

40.13.17 CSEN_DMBASELINE - Delta Modulation Baseline

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	0x0000																0x0000															
Access	RW																RW															
Name	BASELINEDN																BASELINEUP															

Bit	Name	Reset	Access	Description
31:16	BASELINEDN	0x0000	RW	**Delta Modulator Integrator Initial Value**
	When CHOPEN = ENABLE, this field is used to initialize the ramp-down integrator. Unused if CHOPEN = DISABLE.			
15:0	BASELINEUP	0x0000	RW	**Delta Modulator Integrator Initial Value**
	This field is used to initialize the integrator. When CHOPEN = ENABLE, this field is used for the ramp-up integrator.			

40.13.18 CSEN_DMCFG - Delta Modulation Configuration

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				0							0x0		0x0								0x0				0x00							
Access				RW							RW		RW								RW				RW							
Name				DMGRDIS							CRMODE		DMCR								DMR				DMG							

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	DMGRDIS	0	RW	**Delta Modulation Gain Step Reduction Disable**
	If this bit set, the integrator uses a constant gain step for all cycles, given by DMG. Otherwise, at the end of each cycle, the gain step is divided by two and rounded up (DMG = DMG[7:1] + DMG[0]).			
27:22	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
21:20	CRMODE	0x0	RW	**Delta Modulator Conversion Resolution.**
	This field selects the resolution for DM conversions.			

Value	Mode	Description
0	DM10	10-bit delta modulator
1	DM12	12-bit delta modulator
2	DM14	14-bit delta modulator
3	DM16	16-bit delta modulator

19:16	DMCR	0x0	RW	**Delta Modulator Conversion Rate**
	The DMCR field determines how many cycles the DM converter will use per conversion. For the case of DMCR = 0, the converter will perform 16 cycles. Each cycle will consist of between 4 and 64 tests as determined by the DMR field.			
15:12	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
11:8	DMR	0x0	RW	**Delta Modulator Gain Reduction Interval**
	The DMR field determines how many tests are performed and integrated at each gain step setting. For DMR = 0, 64 tests will be performed per cycle (at the same gain step). For DMR = 1 to 15, (DMR x 4) tests will be performed per cycle.			
7:0	DMG	0x00	RW	**Delta Modulator Gain Step**
	This field sets the initial gain step size (the "delta") for the delta modulator. This field represents a number of codes at the resolution selected by CRMODE.			

40.13.19 CSEN_ANACTRL - Analog Control

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				0x7						
Access										RW												RW				RW						
Name										TRSTPROG												IDACIREFS				IREFPROG						

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*		
22:20	TRSTPROG	0x0	RW	**Reset Timing** This field adjusts the amount of time used to discharge the external capacitor during a conversion. Reset timing is increased for larger values of TRSTPROG.
19:11	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
10:8	IDACIREFS	0x0	RW	**Current DAC and Reference Current Scale** This field adjusts the currents used in the CSEN block. This directly affects the drive strength to the external capacitor. For IDACIREFS = 0, the maximum drive strength is used. For IDACIREFS = 1 to 7, the relative drive strength is (IDACIREFS / 8).
7	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
6:4	IREFPROG	0x7	RW	**Reference Current Control.** This field sets the relative magnitude of the current source that charges the internal reference cap. Lower settings allow for larger external capacitance, and higher settings allow for more precise measurements on smaller capacitors.
3:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

40.13.20 CSEN_IF - Interrupt Flag

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																												0	0	0	0	0
Access																												R	R	R	R	R
Name																												APORTCONFLICT	DMAOF	EOS	CONV	CMP

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	APORTCONFLICT	0	R	**APORT Conflict Interrupt Flag**
	1 if any of the BUSes being requested by the CSEN are also being requested by another peripheral			
3	DMAOF	0	R	**DMA Overflow Interrupt Flag.**
	When DMAEN is 1, this flag will be set to 1 by hardware if DMA does not read the DATA register and a new conversion completes.			
2	EOS	0	R	**End of Scan Interrupt Flag.**
	This flag is set to 1 at the end of a scan cycle, after all channels have been converted.			
1	CONV	0	R	**Conversion Done Interrupt Flag**
	This flag is set to 1 when a data conversion is complete and the result has been posted to DATA.			
0	CMP	0	R	**Digital Comparator Interrupt Flag**
	This flag is set to 1 when a CSEN comparator event has happened.			

40.13.21 CSEN_IFS - Interrupt Flag Set

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																												0	0	0	0	0
Access																												W1	W1	W1	W1	W1
Name																												APORTCONFLICT	DMAOF	EOS	CONV	CMP

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	APORTCONFLICT	0	W1	**Set APORTCONFLICT Interrupt Flag**
	Write 1 to set the APORTCONFLICT interrupt flag			
3	DMAOF	0	W1	**Set DMAOF Interrupt Flag**
	Write 1 to set the DMAOF interrupt flag			
2	EOS	0	W1	**Set EOS Interrupt Flag**
	Write 1 to set the EOS interrupt flag			
1	CONV	0	W1	**Set CONV Interrupt Flag**
	Write 1 to set the CONV interrupt flag			
0	CMP	0	W1	**Set CMP Interrupt Flag**
	Write 1 to set the CMP interrupt flag			

40.13.22 CSEN_IFC - Interrupt Flag Clear

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																												0	0	0	0	0
Access																												(R)W1	(R)W1	(R)W1	(R)W1	(R)W1
Name																												APORTCONFLICT	DMAOF	EOS	CONV	CMP

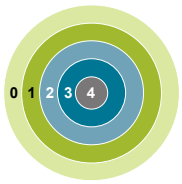
Bit	Name	Reset	Access	Description
31:5	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
4	APORTCONFLICT	0	(R)W1	**Clear APORTCONFLICT Interrupt Flag**
	Write 1 to clear the APORTCONFLICT interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
3	DMAOF	0	(R)W1	**Clear DMAOF Interrupt Flag**
	Write 1 to clear the DMAOF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	EOS	0	(R)W1	**Clear EOS Interrupt Flag**
	Write 1 to clear the EOS interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	CONV	0	(R)W1	**Clear CONV Interrupt Flag**
	Write 1 to clear the CONV interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	CMP	0	(R)W1	**Clear CMP Interrupt Flag**
	Write 1 to clear the CMP interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

40.13.23 CSEN_IEN - Interrupt Enable

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																												0	0	0	0	0
Access																												RW	RW	RW	RW	RW
Name																												APORTCONFLICT	DMAOF	EOS	CONV	CMP

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	APORTCONFLICT	0	RW	**APORTCONFLICT Interrupt Enable**
	Enable/disable the APORTCONFLICT interrupt			
3	DMAOF	0	RW	**DMAOF Interrupt Enable**
	Enable/disable the DMAOF interrupt			
2	EOS	0	RW	**EOS Interrupt Enable**
	Enable/disable the EOS interrupt			
1	CONV	0	RW	**CONV Interrupt Enable**
	Enable/disable the CONV interrupt			
0	CMP	0	RW	**CMP Interrupt Enable**
	Enable/disable the CMP interrupt			

41. CAN - Controller Area Network



Quick Facts

What?

The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed realtime control with a very high level of security.

Why?

The domain of applications for CAN ranges from high speed networks to low cost multiplex wiring. In automotive electronics, engine control units, sensors, anti-skid-systems and more are connected using CAN with bitrates up to 1 Mbit/s. At the same time it is cost effective to build into vehicle body electronics, such as lamp clusters and electric windows, to replace the wiring harness otherwise required.

How?

It provides support for broadcast and multicast communication, with deterministic resolution of contention and robust error detection and signaling.

41.1 Introduction

CAN is a robust multimaster bus supporting a high data transfer rate of 1 Mbit/s and sophisticated error detection and error handling. The CAN peripheral supports up to 32 different message objects for automatic filtering of received messages. Multiple message object locations can be programmed with the same identifier mask to implement receive message FIFOs.

41.2 Features

- Supports CAN protocol version 2.0 part A, B
- Bitrates up to 1 Mbit/s
- Disable Automatic Retransmission mode for Time Triggered CAN applications
- 32 Message Objects
- Each Message Object has its own Identifier Mask
- Programmable FIFO mode
- Maskable interrupt
- Programmable loopback mode for self test operation
- Message RAM retention in EM2

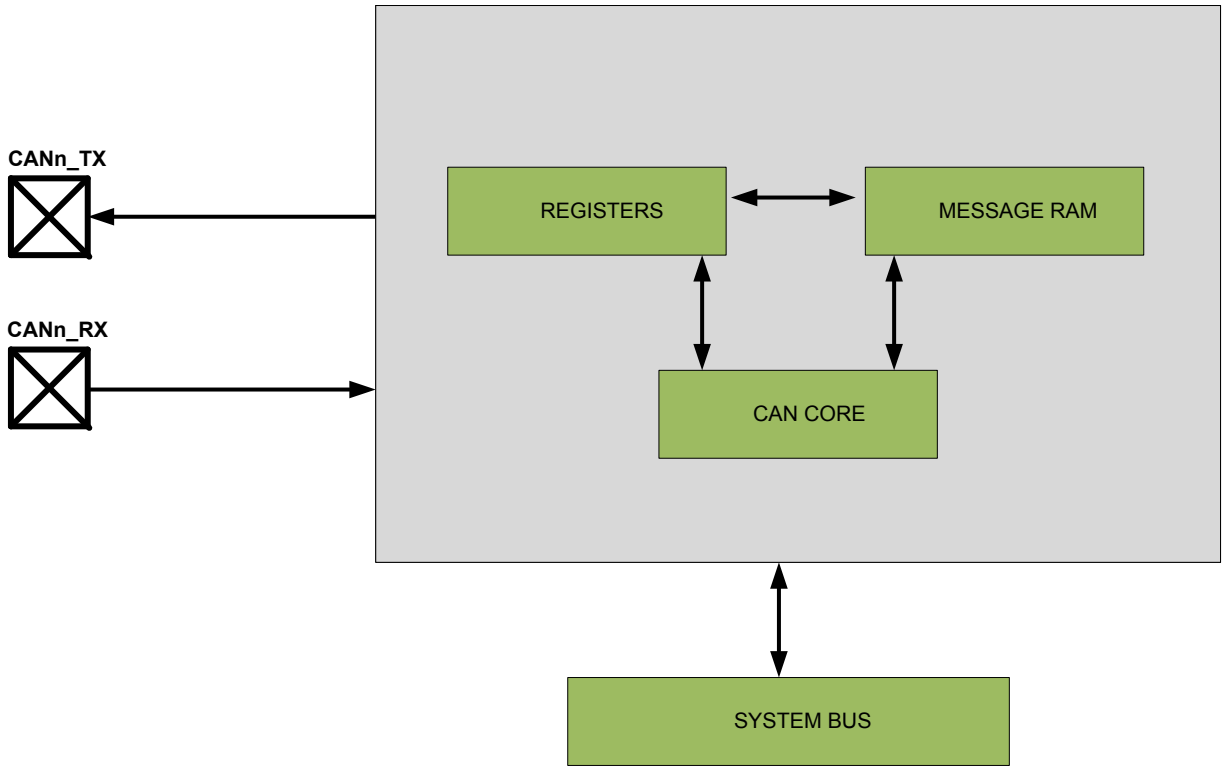


Figure 41.1. Block Diagram

41.3 Functional Description

41.3.1 Operating Modes

41.3.1.1 Software Initialization

The software initialization is started by setting the bit INIT in the CANn_CTRL Register, either by software or by a hardware reset, or by going bus-off.

While INIT is set, all message transfer to and from the CAN bus is stopped, the status of the CAN bus output CAN_TX is recessive (HIGH). The REC and TEC of the CANn_ERRCNT are unchanged. Setting INIT does not change any configuration register.

To initialize the CAN Controller, the CPU has to set up the Bit Timing Register (CANn_BITTIMING) and each Message Object. Message Object is a collective representation of the bits in the Message Interface Registers (CANn_MIRx_CMDMASK, CANn_MIRx_MASK, etc.). More details about the Message Object are stated in 41.3.3 Message Object in the Message Memory . If a Message Object is not needed, it is sufficient to set its MSGVAL bit in CANn_MIRx_ARB to not valid. Otherwise, the whole Message Object has to be initialized.

Access to the Bit Timing Register (CANn_BITTIMING) and to the BRP Extension Register (CANn_BRPE) for the configuration of the bit timing is enabled when both bits INIT and CCE in the CANn_CTRL register are set.

Resetting INIT (by CPU only) finishes the software initialization. Afterwards, the module synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits ("bus idle") before it can take part in bus activities and start the message transfer.

The initialization of the Message Objects is independent of INIT and can be done at runtime, but the Message Objects should all be configured to particular identifiers or set to not valid before the message transfer begins.

To change the configuration of a Message Object during normal operation, the CPU has to start by setting MSGVAL to not valid. When the configuration is completed, MSGVAL is set to valid again.

Table 41.1. Software Initialization

INIT Bit Value	Description
0	Normal Operation
1	Initialization is started

41.3.1.2 CAN Message Transfer

Received messages are stored into their appropriate Message Objects if they pass acceptance filtering. The whole message including all the arbitration bits, DLC and 8 data bytes is stored into the Message Object. If the Identifier Mask is used, the arbitration bits which are masked to “don't care” may be overwritten in the Message Object.

Messages to be transmitted are updated by the CPU. If a permanent Message Object (arbitration and control bits set up during configuration) exists for the message, only the data bytes are updated and then TXRQST bit and DATAVALID bit are set in CANn_MIRx_CTRL to start the transmission. If several transmit messages are assigned to the same Message Object (when the number of Message Objects is not sufficient), the whole Message Object has to be configured before the transmission of this message is requested.

The transmission of any number of Message Objects may be requested at the same time, they are transmitted subsequently according to their internal priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data will be discarded when a message is updated before its pending transmission has started.

Depending on the configuration of the Message Object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

41.3.1.3 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898), the CAN module provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed. By default, this means for automatic retransmission is enabled. It can be disabled to enable the CAN module to work within a Time Triggered CAN (TTCAN, see ISO11898-1) environment.

The Disabled Automatic Retransmission mode is enabled by programming bit DAR in the CANn_CTRL to 1. In this operation mode, the programmer has to consider the different behaviour of the TXRQST and DATAVALID bits in the Control Registers of the Message Buffers (CANn_MIRx_CTRL):

- When a transmission starts, TXRQST of the respective Message Buffer is reset, while DATAVALID remains set.
- When the transmission is completed successfully, the DATAVALID bit is reset.

When a transmission failed (lost arbitration or error) bit DATAVALID remains set. To restart the transmission the CPU has to set TXRQST back to 1.

41.3.1.4 Test Mode

The Test Mode is entered by setting the TEST bit in CANn_CTRL to 1. In Test Mode, the bits TX1, TX0, LBACK, SILENT and BASIC in CANn_TEST are writable. RX monitors the state of pin CAN_RX and therefore is only readable. All Test Register functions are disabled when the bit TEST is reset to 0.

41.3.1.5 Silent Mode

The CAN module can be set in Silent Mode by programming the SILENT bit in CANn_TEST to 1.

In Silent Mode, the CAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN core is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN module monitors this dominant bit, although the CAN bus may remain in recessive state. The Silent Mode can be used to analyse the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). Figure 41.2 CAN Core in Silent Mode on page 1781 shows the connection of signals CAN_TX and CAN_RX to the CAN core in Silent Mode.

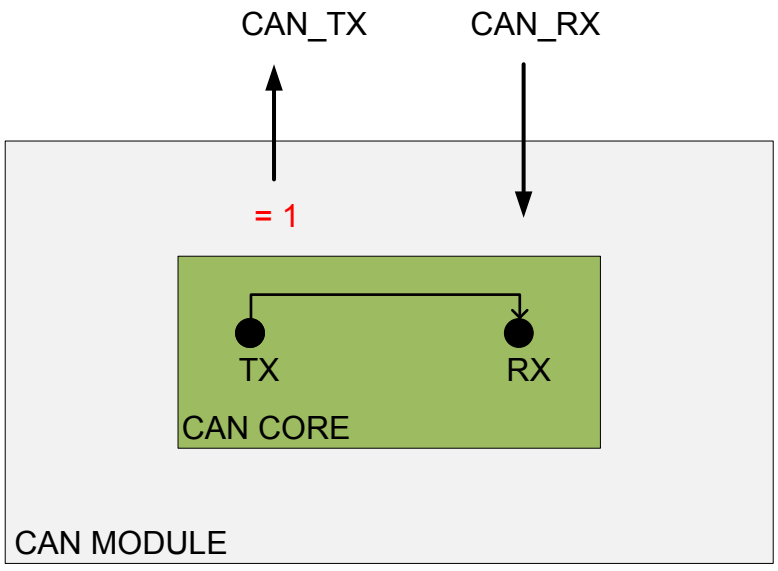


Figure 41.2. CAN Core in Silent Mode

41.3.1.6 Loop Back Mode

The CAN module can be set in Loop Back Mode by programming the CANn_TEST bit LBACK to 1 . In Loop Back Mode, the module treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into a Receive Buffer. Figure 41.3 CAN Core in Loop Back Mode on page 1782 shows the connection of signals CAN_TX and CAN_RX to the CAN core in Loop Back Mode.

This mode is provided for self-test functions. To be independent from external stimulation, the CAN core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/ remote frame) in Loop Back Mode. In this mode the CAN core performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN_RX input pin is disregarded by the CAN core. The transmitted messages can be monitored at the CAN_TX pin.

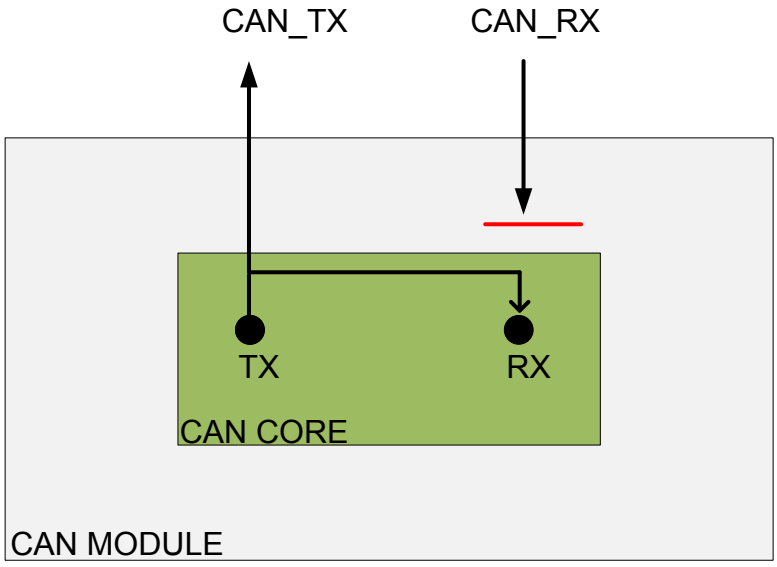


Figure 41.3. CAN Core in Loop Back Mode

41.3.1.7 Loop Back Combined With Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by programming bits LBACK and SILENT to 1 at the same time. This mode can be used for a “Hot Selftest”, meaning the CAN can be tested without affecting a running CAN system connected to the pins CAN_TX and CAN_RX. In this mode the CAN_RX pin is disconnected from the CAN core and the CAN_TX pin is held recessive. Figure 41.4 CAN Core in Loop Back Combined With Silent Mode on page 1783 shows the connection of signals CAN_TX and CAN_RX to the CAN core in case of the combination of Loop Back Mode with Silent Mode.

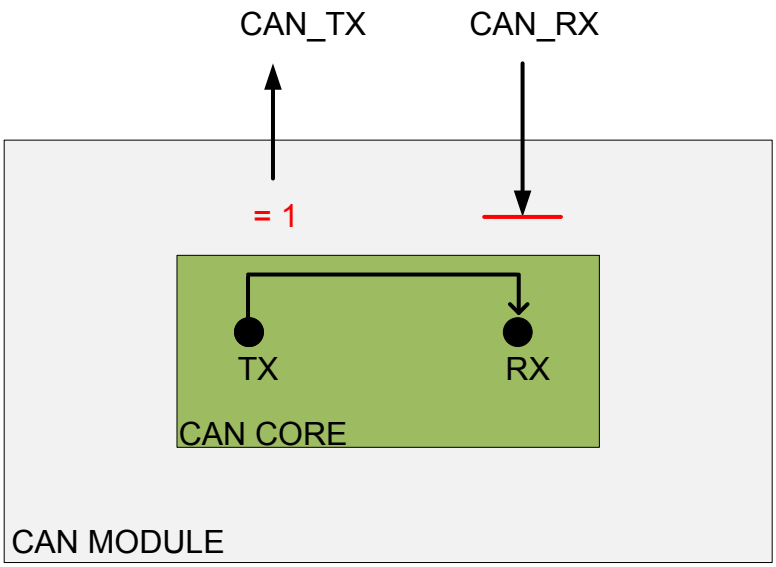


Figure 41.4. CAN Core in Loop Back Combined With Silent Mode

41.3.1.8 Basic Mode

The CAN module can be set in Basic Mode by programming the CANn_TEST bit BASIC to 1

The CANn_MIR0 Registers are used as Transmit Buffer. The transmission of the contents of the CANn_MIR0 Registers is requested by writing the BUSY bit of the CANn_MIR0 Command Request Register to 1. The CANn_MIR0 Registers are locked while the BUSY bit is set. The BUSY bit indicates that the transmission is pending.

As soon the CAN bus is idle, the CANn_MIR0 Registers are loaded into the shift register of the CAN core and the transmission is started. When the transmission has completed, the BUSY bit is reset and the locked CANn_MIR0 Registers are released.

A pending transmission can be aborted at any time by resetting the BUSY bit in the CANn_MIR0 Command Request Register while the CANn_MIR0 Registers are locked. If the CPU has reset the BUSY bit, a possible retransmission in case of lost arbitration or in case of an error is disabled.

The CANn_MIR1 Registers are used as Receive Buffer. After the reception of a message the contents of the shift register is stored into the CANn_MIR1 Registers, without any acceptance filtering.

Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read Message Object is initiated by writing the BUSY bit of the CANn_MIR1_CMDREQ Register to 1, the contents of the shift register is stored into the CANn_MIR1 Registers.

In basic mode the evaluation of all Message Object related control and status bits and of the control bits of the CANn_MIRx_CMDMASK Registers is turned off. The message number of the CANn_MIRx_CMDREQ registers is not evaluated. The DATAVALID and MESSAGEOF bits of the CANn_MIR1_CTRL Register retain their function, DLC3-0 will show the received DLC, the other control bits will be read as 0.

41.3.1.9 Software Control of Pin CAN_TX

4 output functions are available for the CAN transmit pin CAN_TX. Additionally to its default function -- the serial data output -- it can drive the CAN Sample Point signal to monitor the CAN core's bit timing and it can drive constant dominant or recessive values. The last two functions, combined with the readable CAN receive pin CAN_RX, can be used to check the CAN bus physical layer.

The output mode of the CAN_TX pin is selected by programming the CANn_TEST bits TX1 and TX0

The three test functions for the CAN_TX pin interfere with all CAN protocol functions. CAN_TX must be left in its default function when CAN message transfer or any of the test modes Loop Back Mode, Silent Mode, or Basic Mode are selected.

41.3.2 Message Interface Register Sets

There are two sets of Interface Registers (CANn_MIRx) which are used to control the CPU access to the Message RAM. The Interface Registers avoid conficts between CPU access to the Message RAM and CAN message reception and transmission by buffering the data to be transferred. A complete Message Object or parts of the Message Object may be transferred between the Message RAM and the CANn_MIRx Message Buffer registers in one single transfer.

The function of the two interface register sets is identical (except for test mode BASIC). They can be used the way that one set of registers is used for data transfer to the Message RAM while the other set of registers is used for the data transfer from the Message RAM, allowing both processes to be interrupted by each other.

Each set of Interface Registers consists of Message Buffer Registers (CANn_MIRx_DATAL and CANn_MIRx_DATAH) controlled by their own Command Registers (CANn_MIRx_CTRL). The Command Mask Register (CANn_MIRx_CMDMASK) specifies the direction of the data transfer and which parts of a Message Object will be transferred. The Command Request Register (CANn_MIRx_CMDREQ) is used to select a Message Object in the Message RAM as target or source for the transfer and to start the action specified in the Command Mask Register (CANn_MIRx_CMDMASK).

Table 41.2. Interface Register Sets

Interface Registers 0	Address	Interface Registers 1	Address
CANn_BASE + 0x60	MIR0 Command Mask	CANn_BASE + 0x80	MIR1 Command Mask
CANn_BASE + 0x64	MIR0 Mask	CANn_BASE + 0x84	MIR1 Mask
CANn_BASE + 0x68	MIR0 Arbitration	CANn_BASE + 0x88	MIR1 Arbitration
CANn_BASE + 0x6C	MIR0 Message Control	CANn_BASE + 0x8C	MIR1 Message Control
CANn_BASE + 0x70	MIR0 DATA L	CANn_BASE + 0x90	MIR1 DATA L
CANn_BASE + 0x74	MIR0 DATA H	CANn_BASE + 0x94	MIR1 DATA H
CANn_BASE + 0x78	MIR0 Command Request	CANn_BASE + 0x98	MIR1 Command Request

41.3.3 Message Object in the Message Memory

There are 32 Message Objects in the Message RAM. To avoid conflicts between CPU access to the Message RAM and CAN message reception and transmission, the CPU cannot directly access the Message Objects, these accesses are handled via the CANn_MIRx Interface Registers. The bit definitions can be found in descriptions of the CANn_MIRx registers.

Table 41.3. Message Object

UMASK	MSK28-0	MXTD	MDIR	EOB	DATA-VALID	MESSA-GEOF	RXIE	TXIE	INTPND	RMTEN	TXRQST	N.A.
MSGVAL	ID28-0	XTD	DIR	DLC3-0	DATA0	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7

41.3.4 Management of Message Objects

All the Message Objects must be initialized by the CPU or they must be not valid (MSGVAL = 0) and the bit timing must be configured before the CPU clears the INIT bit in the CANn_CTRL Register.

The configuration of a Message Object is done by programming Mask, Arbitration, Control and Data field of one of the two interface register sets to the desired values. By writing to the corresponding CANn_MIRx_CMDREQ Register, the CANn_MIRx_DATAL and CANn_MIRx_DATAH Registers are loaded into the addressed Message Object in the Message RAM.

The CPU reads received messages and updates messages to be transmitted via the CANn_MIRx Interface Registers. Depending on the configuration, the CPU is interrupted on certain CAN message and CAN error events.

41.3.5 Data Transfer From/to Message RAM

When the CPU initiates a data transfer between the CANn_MIRx Registers and Message RAM, the Message Handler sets the BUSY bit in the respective CANn_MIRx_CMDREQ to 1. After the transfer has completed, the BUSY bit is set back to 0.

The respective Command Mask Register CANn_MIRx_CMDMASK specifies whether a complete Message Object or only parts of it will be transferred. Due to the structure of the Message RAM it is not possible to write single bits/bytes of one Message Object, it is always necessary to write a complete Message Object into the Message RAM. Therefore the data transfer from the CANn_MIRx Registers to the Message RAM requires of a read-modify-write cycle. First that parts of the Message Object that are not to be changed are read from the Message RAM and then the complete contents of the Message Buffer (CANn_MIRx_DATAL and CANn_MIRx_DATAH) Registers are transferred into the Message Object.

After the partial write of a Message Object, that Message Buffer (CANn_MIRx_DATAL and CANn_MIRx_DATAH) Registers that are not selected in the Command Mask CANn_MIRx_CMDMASK Register will set to the actual contents of the selected Message Object.

After the partial read of a Message Object, that Message Buffer (CANn_MIRx_DATAL and CANn_MIRx_DATAH) Registers that are not selected in the Command Mask CANn_MIRx_CMDMASK Register will be left unchanged.

41.3.6 Transmission of Messages

If the shift register of the CAN is ready for loading and if there is no data transfer between the CANn_MIRx Registers and Message RAM, the VALID bits in the CANn_MESSAGESTATE Register and TXRQSTOUT bits in the CANn_TRANSREQ Register are evaluated. The valid Message Object with the highest priority pending transmission request is loaded into the shift register and the transmission is started. The Message Object's DATAVALID bit is reset.

After a successful transmission and if no new data was written to the Message Object (DATAVALID = 0) since the start of the transmission, the TXRQST bit will be reset. If TXIE in CANn_MIRx_CTRL is set, INTPND in CANn_MIRx_CTRL will be set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

41.3.7 Acceptance Filtering of Received Messages

When the arbitration and control field (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the Rx/Tx Shift Register of the CAN module, it starts the scanning of the Message RAM for a matching valid Message Object.

To scan the Message RAM for a matching Message Object, the arbitration bits from the CAN shift register and then the arbitration and mask fields (including MSGVAL, UMASK, DATAVALID, and EOB) of Message Object 1 are compared. This is repeated with each following Message Object until a matching Message Object is found or until the end of the Message RAM is reached

If a match occurs, the scanning is stopped and the Module proceeds depending on the type of frame (Data Frame or Remote Frame) received.

41.3.7.1 Reception of Data Frame

Not only the data bytes, but all arbitration bits and the Data Length Code are stored into the corresponding Message Object. This is implemented to keep the data bytes connected with the identifier even if arbitration mask registers are used.

The DATAVALID bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset DATAVALID when it reads the Message Object. If at the time of the reception the DATAVALID bit was already set, MESSAGEOF is set to indicate that the previous data (supposedly not seen by the CPU) is lost. If the RXIE bit is set, the INTPND bit is set, causing the Interrupt Register CANn_INTID to point to this Message Object.

The TXRQST bit of this Message Object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

41.3.7.2 Reception of Remote Frame

When a Remote Frame is received, 3 difference configurations of the matching Message Object have to be considered:

- DIR = 1 (direction = transmit), RMTEN = 1, UMASK = 1 or 0. At the reception of a matching Remote Frame, the TXRQST bit of this Message Object is set. The rest of the Message Object remains unchanged
- DIR = 1 (direction = transmit), RMTEN = 0, UMASK = 0. At the reception of a matching Remote Frame, the TXRQST bit of this Message Object remains unchanged; the Remote Frame is ignored.
- DIR = 1 (direction = transmit), RMTEN = 0, UMASK = 1. At the reception of a matching Remote Frame, the TXRQST bit of this Message Object is reset. The arbitration and control field (Identifier + IDE + RTR + DLC) from the shift register is stored into the Message Object in the Message RAM and the DATVALID bit of this Message Object is set. The data fifield of the Message Object remains unchanged; the Remote Frame is treated similar to a received Data Frame.

41.3.8 Receive/Transmit Priority

The receive/transmit priority for the Message Objects is attached to the message number. Message Object 1 has the highest priority, while Message Object 32 has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding Message Object.

41.3.9 Configuration of a Transmit Object

Table 41.4. Configuration of Transmit Object

MSGVAL	ARB	DATA	MASK	EOB	DIR	DATA-VALID	MESSA-GEOF	RXIE	TXIE	INTPND	RMTEN	TXRQST
1	Applica-tion	Applica-tion	Applica-tion	1	1	0	0	0	Applica-tion	0	Applica-tion	0

The Arbitration Registers (ID28-0 and XTD bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID28 - ID18, ID17 - ID0 can then be disregarded

If the RMTEN bit is set, a matching received Remote Frame will cause the TXRQST bit to be set; the Remote Frame will autonomously be answered by a Data Frame

The Data Registers (DLC3-0, Data0-7) are given by the application, TXRQST and RMTEN may not be set before the data is valid

The Mask Registers (MASK28-0, UMASK, MXTD, and MDIR bits) may be used (UMASK=’1’) to allow groups of Remote Frames with similar identifiers to set the TXRQST bit.

41.3.10 Updating a Transmit Object

The CPU may update the data bytes (DATA7:0) of a Transmit Object any time via the CANn_MIRx Interface registers, neither MSGVAL nor TXRQST have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding CANn_MIRx_DATAL Register or MIRx CANn_MIRx_DATAH Register have to be valid before the content of that register is transferred to the Message Object. Either the CPU has to write all four bytes into the MIRx Data Register or the Message Object is transferred to the MIRx Data Register before the CPU writes the new data bytes.

When only the (eight) data bytes are updated, first 0x0087 is written to the Command Mask Register and then the number of the Message Object is written to the Command Request Register, concurrently updating the data bytes and setting TXRQST.

To prevent the reset of TXRQST at the end of a transmission that may already be in progress while the data is updated, DATAVALID has to be set together with TXRQST.

When DATAVALID is set together with TXRQST, DATAVALID will be reset as soon as the new transmission has started.

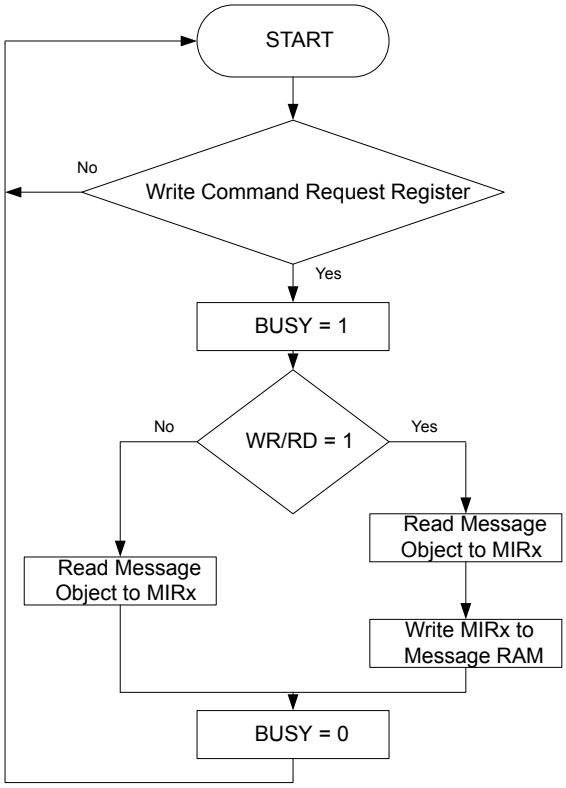


Figure 41.5. Data Transfer Between MIRx Registers and the Message RAM

41.3.11 Configuration of a Receive Object

Table 41.5. Configuration of Receive Object

MSGVAL	ARB	DATA	MASK	EOB	DIR	DATA-VALID	MESSA-GEOF	RXIE	TXIE	INTPND	RMTEN	TXRQST
1	Applica-tion	Applica-tion	Applica-tion	1	0	0	0	Applica-tion	0	0	0	0

The Arbitration Registers (ID28-0 and XTD bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID28 - ID18, ID17 - ID0 can then be disregarded. When a Data Frame with an 11-bit Identifier is received, ID17 - ID0 will be set to ‘0.’

If the RxIE bit is set, the INTPND bit will be set when a received Data Frame is accepted and stored in the Message Object.

The Data Length Code (DLC3-0) is given by the application. When the Message Handler stores a Data Frame in the Message Object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by non specified values.

The Mask Registers (MASK28-0, UMASK, MXTD, and MDIR bits) may be used (UMask=’1’) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications.

41.3.12 Handling of Received Messages

The CPU may read a received message any time via the CANn_MIRx Interface registers.

Typically the CPU will write first 0x007F to the Command Mask Register and then the number of the Message Object to the Command Request Register. That combination will transfer the whole received message from the Message RAM into the Message Buffer Register. Additionally, the bits DATAVALID and INTPND are cleared in the Message RAM (not in the Message Buffer).

If the Message Object uses masks for acceptance filtering, the arbitration bits show which of the matching messages has been received.

The actual value of DATAVALID shows whether a new message has been received since last time this Message Object was read. The actual value of MESSAGEOF shows whether more than one message has been received since last time this Message Object was read. MESSAGEOF will not be automatically reset.

By means of a Remote Frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TXRQST bit of a receive object will cause the transmission of a Remote Frame with the receive object’s identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TXRQST bit is automatically reset.

41.3.13 Configuration of a FIFO Buffer

With the exception of the EOB bit, the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object.

To concatenate two or more Message Objects into a FIFO Buffer, the identifiers and masks (if used) of these Message Objects have to be programmed to matching values. Due to the implicit priority of the Message Objects, the Message Object with the lowest number will be the first Message Object of the FIFO Buffer. The EOB bit of all Message Objects of a FIFO Buffer except the last have to be programmed to 0. The EOB bits of the last Message Object of a FIFO Buffer is set to 1, configuring it as the End of the Block.

41.3.14 Reception of Messages With FIFO Buffers

Received messages with identifiers matching to a FIFO Buffer are stored into a Message Object of this FIFO Buffer starting with the Message Object with the lowest message number.

When a message is stored into a Message Object of a FIFO Buffer the DATAVALID bit of this Message Object is set. By setting DATAVALID while EOB is 0 the Message Object is locked for further write accesses by the Message Handler until the CPU has written the DATAVALID bit back to 0.

Messages are stored into a FIFO Buffer until the last Message Object of this FIFO Buffer is reached. If none of the preceding Message Objects is released by writing DATAVALID to 0, all further messages for this FIFO Buffer will be written into the last Message Object of the FIFO Buffer and will therefore overwrite previous messages.

41.3.15 Reading From a FIFO Buffer

When the CPU transfers the contents of Message Object to the CANn_MIRx Message Buffer registers by writing its number to the MIRx Command Request Register, the corresponding Command Mask Register should be programmed the way that bits DATAVALID and INTPND are reset to 0 (TXRQST/DATAVALID = 1 and CLRINTPND = 1). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

To assure the correct function of a FIFO Buffer, the CPU should read out the Message Objects starting at the FIFO Object with the lowest message number.

41.3.16 Handling of Interrupts

If several interrupts are pending, the CAN Interrupt Identification Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it.

The Status Interrupt has the highest priority. Among the message interrupts, the Message Object's interrupt priority decreases with increasing message number.

A message interrupt is cleared by clearing the Message Object's INTPND bit. The Status Interrupt is cleared by reading the CANn_STATUS Register or by using the CANn_IF0IFC and CANn_IF1IFC registers respectively.

The interrupt identifier INTID in the CANn_INTID Register indicates the cause of the interrupt. When no interrupt is pending, the register will hold the value 0. If the value of the CANn_INTID Register is different from 0, then there is an interrupt pending and, if IE is set, the interrupt line to the CPU, is active (unless the corresponding bits in the CANn_IF0IEN and CANn_IF1IEN are 0). The interrupt line remains active until the CANn_INTID is back to value zero (the cause of the interrupt is reset) or until IE is reset.

The value 0x8000 indicates that an interrupt is pending because the CAN has updated (not necessarily changed) the Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits RXOK, TXOK and LEC, but a write access of the CPU to the Status Register can never generate or reset an interrupt.

All other values indicate that the source of the interrupt is one of the Message Objects, INTID points to the pending message interrupt with the highest interrupt priority.

The CPU controls whether a change of the Status Register may cause an interrupt (bits EIE and SIE in the CAN Control Register) and whether the interrupt line becomes active when the Interrupt Identification Register is different from zero (the IF1IEN register and the bit IE in the CAN Control Register). The Interrupt Indentification Register will be updated even when IE is reset

The CPU has 3 possibilities to follow the source of a message interrupt. First, it can follow the INTID in the Interrupt Identification Register. Second, it can poll the Interrupt Pending Register and finally it can read the CANn_IF0IF register.

An interrupt service routine reading the message that is the source of the interrupt may read the message and reset the Message Object's INTPND at the same time (bit CLRINTPND in the Command Mask Register). When INTPND is cleared, the CANn_INTID will point to the next Message Object with a pending interrupt.

41.3.17 Configuration of the Bit Timing

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly. In many cases, the CAN bit synchronisation will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

Table 41.6. Parameters of the CAN Bit Time

Parameter	Range	Remark
BRP	[1..32]	Defines the length of the time quantum t_q
Sync_Seg	1 t_q	fixed length, synchronization of bus input to the system clock
Prop_Seg	[1..8] t_q	compensates for the physical delay times
Phase_Seg1	[1..8] t_q	may be lengthened temporarily by synchronization
Phase_Seg2	[1..8] t_q	may be shortenend temporarily by synchronization
SJW	[1..4] t_q	may not be longer than either Phase Buffer Segment

41.3.17.1 Bit Time and Bit Rate

CAN supports bit rates in the range of lower than 1 kBit/s up to 1000 kBit/s. Each member of the CAN network has its own clock generator, usually a quartz oscillator. The timing parameter of the bit time (i.e. the reciprocal of the bit rate) can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods osc may be different.

According to the CAN specification, the bit time is divided into four segments. The Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's system clock f_{sys} and the Baud Rate Prescaler (BRP): $t_q = BRP / f_{sys}$.

The Synchronisation Segment Sync_Seg is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync_Seg and the Sync_Seg is called the phase error of that edge. The Propagation Time Segment (Prop_Seg) is intended to compensate for the physical delay times within the CAN network. The Phase Buffer Segments (Phase_Seg1) and (Phase_Seg2) surround the Sample Point. The (Re-)Synchronisation Jump Width (SJW) defines how far a resynchronisation may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

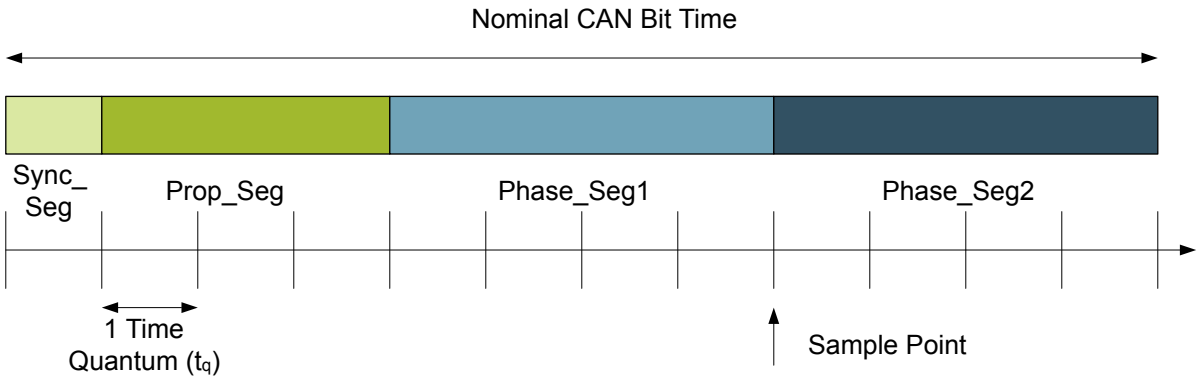


Figure 41.6. Bit Timing

41.3.17.2 EM2 Operation

The CAN module is retained in EM2 (including the Message RAM). However, to ensure that the Message RAM data is not corrupted, the INIT bit in the Control Register needs to be 1 before entering EM2. This will mean that the user will have to clear the INIT upon exit from EM2 so that the CAN module can resynchronize itself to the bus and then take part in the sending and receiving of messages.

41.3.17.3 Software BIST for Message RAM

The Message RAM is tested using a software BIST. The CANn_MDATA and tbe CANn_MEMACC registers are present for this purpose. The CANn_MDATA register is used to hold the read/write data from/to the RAM and CANn_MEMACC is used to access a particular memory location. It should be noted, that the Message RAM is 32 locations deep and 136 bits wide so to exercise a location in its enitrety the SUBWORD field needs to be used (since access is restricted to word boundaries).

41.4 Register Map

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

Offset	Name	Type	Description
0x000	CANn_CTRL	RW	Control Register
0x004	CANn_STATUS	RWH	Status Register
0x008	CANn_ERRCNT	R	Error Count Register
0x00C	CANn_BITTIMING	RW	Bit Timing Register
0x010	CANn_INTID	R	Interrupt Identification Register
0x014	CANn_TEST	RWH	Test Register
0x018	CANn_BRPE	RW	BRP Extension Register
0x01C	CANn_TRANSREQ	R	Transmission Request Register
0x020	CANn_MESSAGEDATA	R	New Data Register
0x028	CANn_MESSAGESTATE	R	Message Valid Register
0x02C	CANn_CONFIG	RW	Configuration Register
0x030	CANn_IF0IF	R	Message Object Interrupt Flag Register
0x034	CANn_IF0IFS	W1	Message Object Interrupt Flag Set Register
0x038	CANn_IF0IFC	(R)W1	Message Object Interrupt Flag Clear Register
0x03C	CANn_IF0IEN	RW	Message Object Interrupt Enable Register
0x040	CANn_IF1IF	R	Status Interrupt Flag Register
0x044	CANn_IF1IFS	W1	Message Object Interrupt Flag Set Register
0x048	CANn_IF1IFC	(R)W1	Message Object Interrupt Flag Clear Register
0x04C	CANn_IF1IEN	RW	Status Interrupt Enable Register
0x050	CANn_ROUTE	RW	I/O Routing Register
0x060	CANn_MIR0_CMDMASK	RW	Interface Command Mask Register
0x064	CANn_MIR0_MASK	RW	Interface Mask Register
0x068	CANn_MIR0_ARB	RW	Interface Arbitration Register
0x06C	CANn_MIR0_CTRL	RWH	Interface Message Control Register
0x070	CANn_MIR0_DATAL	RW	Interface Data a Register
0x074	CANn_MIR0_DATAH	RW	Interface Data B Register
0x078	CANn_MIR0_CMDREQ	RWH	Interface Command Request Register
0x080	CANn_MIR1_CMDMASK	RW	Interface Command Mask Register
0x084	CANn_MIR1_MASK	RW	Interface Mask Register
0x088	CANn_MIR1_ARB	RW	Interface Arbitration Register
0x08C	CANn_MIR1_CTRL	RWH	Interface Message Control Register
0x090	CANn_MIR1_DATAL	RW	Interface Data a Register
0x094	CANn_MIR1_DATAH	RW	Interface Data B Register
0x098	CANn_MIR1_CMDREQ	RWH	Interface Command Request Register

41.5 Register Description

41.5.1 CANn_CTRL - 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																									0	0	0		0	0	0	1
Access																									RW	RW	RW		RW	RW	RW	RW
Name																									TEST	CCE	DAR		EIE	SIE	IE	INIT

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	TEST	0	RW	**Test Mode Enable Write** Enables access to the Test Register.
6	CCE	0	RW	**Configuration Change Enable** Enables access to the Bit Timing Register
5	DAR	0	RW	**Disable Automatic Retransmission** When set, automatic retransmission is disabled.
4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	EIE	0	RW	**Error Interrupt Enable** Enables error interrupts on status register change.
2	SIE	0	RW	**Status Change Interrupt Enable** Enables status interrupts on status register change.
1	IE	0	RW	**Module Interrupt Enable** Top level interrupt enable for the CAN module.
0	INIT	1	RW	**Initialize** The Bus Off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting INIT. If the device goes Bus Off, it will set INIT of its own accord, stopping all bus activities. Once INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operations. At the end of the Bus Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the Status Register, enabling the CPU to readily check whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the Bus Off recovery sequence.

41.5.2 CANn_STATUS - Status 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																									0	0	0	0	0	0x0		
Access																									R	R	R	RW	RW	RW		
Name																									BOFF	EWARN	EPASS	RXOK	TXOK	LEC		

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	BOFF	0	R	**Bus Off Status**
6	EWARN	0	R	**Warning Status**
5	EPASS	0	R	**Error Passive**
4	RXOK	0	RW	**Received a Message Successfully**
3	TXOK	0	RW	**Transmitted a Message Successfully**
2:0	LEC	0x0	RW	**Last Error Code**

The LEC field holds a code which indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. The unused code 0x07 may be written by the CPU to check for updates.

Value	Mode	Description
0	NONE	No error occurred during last CAN bus event.
1	STUFF	More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
2	FORM	A fixed format part of a received frame has the wrong format.
3	ACK	The message this CAN Core transmitted was not acknowledged by another node.
4	BIT1	During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
5	BIT0	During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored Bus value was recessive. During Bus Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

Bit	Name	Reset	Access	Description
	6	CRC		The CRC check sum was incorrect in the message received; the CRC received for an incoming message does not match with the calculated CRC for the received data.
	7	UNUSED		When the LEC shows the value '7', no CAN bus event was detected since the CPU wrote this value to the LEC.

41.5.3 CANn_ERRCNT - Error Count 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																	0	0x00							0x00							
Access																	R	R							R							
Name																	RECERRP	REC							TEC							

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	RECERRP	0	R	**Receive Error Passive**

Value	Mode	Description
0	FALSE	The Receive Error Counter is below the error passive level.
1	TRUE	The Receive Error Counter has reached the error passive level as defined in the CAN Specification.

14:8	REC	0x00	R	**Receive Error Counter**
	Actual state of the Receive Error Counter. Values between 0 and 127.			
7:0	TEC	0x00	R	**Transmit Error Counter**
	Actual state of the Transmit Error Counter. Values between 0 and 255.			

41.5.4 CANn_BITTIMING - Bit Timing 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																		0x2			0x3				0x0		0x01					
Access																		RW			RW				RW		RW					
Name																		TSEG2			TSEG1				SJW		BRP					

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*		
14:12	TSEG2	0x2	RW	**Time Segment After the Sample Point**
	The valid values for TSEG2 are [0.....7]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.			
11:8	TSEG1	0x3	RW	**Time Segment Before the Sample Point**
	The valid values for TSEG1 are [1.....15]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.			
7:6	SJW	0x0	RW	**Synchronization Jump Width**
	The valid programmed values are [0...3]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.			
5:0	BRP	0x01	RW	**Baud Rate Prescaler**
	The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of the quanta. Valid values for the Baud Rate Prescaler are [0...63]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.			

41.5.5 CANn_INTID - Interrupt Identification 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																	0										0x00					
Access																	R										R					
Name																	INTSTAT										INTID					

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	INTSTAT	0	R	**Status Interupt**
	A Status Interrupt is generated by bits BOFF and EWARN (Error Interrupt) or by RXOK, TXOK, and LEC (Status Change Interrupt) assumed that the corresponding enable bits in the CAN Control Register are set. A change of bit EPASS or a write to RXOK, TXOK, or LEC will never generate a Status Interrupt. Reading the Status Register will clear the Status Interrupt value (0x8000) in the Interrupt Identification Register, if it is pending.			
14:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	INTID	0x00	R	**Interrupt Identifier**
	Number here indicated the source of the interrupt.0: No interrupt is pending. 1-32: Number of Message Object which caused the interrupt. 33-63: Unused.			

Value	Mode	Description
0	FALSE	Status Interrupt is cleared
1	TRUE	Status Interrupt is generated

41.5.6 CANn_TEST - Test 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																									0	0x0		0	0	0		
Access																									R	RW		RW	RW	RW		
Name																									RX	TX		LBACK	SILENT	BASIC		

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	RX	0	R	**Monitors the Actual Value of CAN_RX Pin**
	Write a 1 to this bit to start timer.			
	Value	Mode		Description
	0	LOW		CAN bus is dominant.
	1	HIGH		CAN bus is recessive.
6:5	TX	0x0	RW	**Control of CAN_TX Pin**
	The different test functions may be combined, but tx[1:0] not equal to 0 disturbs message transfer.			
	Value	Mode		Description
	0	CORE		Reset value, CAN_TX is controlled by the CAN Core.
	1	SAMPT		Sample Point can be monitored at CAN_TX pin.
	2	LOW		CAN_TX pin drives a dominant bit (0) value.
	3	HIGH		CAN_TX pin drives a recessive bit (1) value.
4	LBACK	0	RW	**Loopback Mode**
	When set, CAN treats its own transmitted messages as received messages.			
3	SILENT	0	RW	**Silent Mode**
	When set, CAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission.			
2	BASIC	0	RW	**Basic Mode**
	Enables low-level data transmit and receive via CANn_MIR0_xxx and CANn_MIR1_xxx registers.			
1:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

41.5.7 CANn_BRPE - BRP Extension 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																													BRPE		

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	BRPE	0x0	RW	**Baud Rate Prescaler Extension**
	By programming BRPE the Baud Rate Prescaler can be extended to values up to 1023. The actual interpretation by the hardware is that one more than the value programmed by BRPE (MSBs) and BRP (LSBs) is used.			

41.5.8 CANn_TRANSREQ - Transmission Request 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																0x00000000																
Access																R																
Name																TXRQSTOUT																

Bit	Name	Reset	Access	Description
31:0	TXRQSTOUT	0x00000000	R	**Transmission Request Bits (Of All Message Objects)**

By reading the TXRQSTOUT bits, the CPU can check for which Message Object's Transmission Request is pending.

Value	Mode	Description
0	FALSE	This Message Object is not waiting for transmission.
1	TRUE	The transmission of this Message Object is requested and is not yet done.

41.5.9 CANn_MESSAGEDATA - New 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	0x00000000																															
Access	R																															
Name	VALID																															

Bit	Name	Reset	Access	Description
31:0	VALID	0x00000000	R	**DATAVALID Bits (of All Message Objects)**
	By reading out the VALID bits, the CPU can check for which Message Object the data portion was updated.			

41.5.10 CANn_MESSAGESTATE - Message Valid 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	0x00000000																															
Access	R																															
Name	VALID																															

Bit	Name	Reset	Access	Description
31:0	VALID	0x00000000	R	**Message Valid Bits (of All Message Objects)**
	By reading out the VALID bits, the CPU can check which Message Object is valid.			

41.5.11 CANn_CONFIG - Configuration 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																	0															
Access																	RW															
Name																	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	DBGHALT	0	RW	**Debug Halt**

Value	Mode	Description
0	NORMAL	Normal operation when debug mode is active
1	STALL	Stall when debug mode is active. Register write access is blocked in this mode

14:0	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

41.5.12 CANn_IF0IF - Message Object Interrupt Flag 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																	0x00000000															
Access																	R															
Name																	MESSAGE															

Bit	Name	Reset	Access	Description
31:0	MESSAGE	0x00000000	R	**Message Object Interrupt Flag**

41.5.13 CANn_IF0IFS - Message Object Interrupt Flag Set 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	0x00000000																															
Access	W1																															
Name	MESSAGE																															

Bit	Name	Reset	Access	Description
31:0	MESSAGE	0x00000000	W1	**Set MESSAGE Interrupt Flag**
	Write 1 to set the MESSAGE interrupt flag			

41.5.14 CANn_IF0IFC - Message Object Interrupt Flag Clear 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	0x00000000																															
Access	(R)W1																															
Name	MESSAGE																															

Bit	Name	Reset	Access	Description
31:0	MESSAGE	0x00000000	(R)W1	**Clear MESSAGE Interrupt Flag**
	Write 1 to clear the MESSAGE interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

41.5.15 CANn_IF0IEN - Message Object Interrupt Enable 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	0xFFFFFFFF																															
Access	RW																															
Name	MESSAGE																															

Bit	Name	Reset	Access	Description
31:0	MESSAGE	0xFFFFFFFF	RW	**MESSAGE Interrupt Enable**
	Enable/disable the MESSAGE interrupt			

41.5.16 CANn_IF1IF - Status 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																																0
Access																																R
Name																																STATUS

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	STATUS	0	R	**Status Interrupt Flag**

41.5.17 CANn_IF1IFS - Message Object Interrupt Flag Set 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																																0
Access																																W1
Name																																STATUS

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	STATUS	0	W1	**Set STATUS Interrupt Flag**
	Write 1 to set the STATUS interrupt flag			

41.5.18 CANn_IF1IFC - Message Object Interrupt Flag Clear 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																																0
Access																																(R)W1
Name																																STATUS

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	STATUS	0	(R)W1	**Clear STATUS Interrupt Flag**
	Write 1 to clear the STATUS interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

41.5.19 CANn_IF1IEN - Status 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																																1
Access																																RW
Name																																STATUS

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	STATUS	1	RW	**STATUS Interrupt Enable**
	Enable/disable the STATUS interrupt			

41.5.20 CANn_ROUTE - I/O Routing 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																			0x00						0x00							0
Access																			RW						RW							RW
Name																			TXLOC						RXLOC							TXPEN

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*		
13:8	TXLOC	0x00	RW	**TX Pin Location**

Decides the location of the CAN_TX pin .

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6

7:2	RXLOC	0x00	RW	**RX Pin Location**

Decides the location of the CAN_RX pin .

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4
5	LOC5	Location 5
6	LOC6	Location 6

1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	TXPEN	0	RW	**TX Pin Enable**

When Enabled the CAN_TX pin is enabled.

Value	Description
0	The CAN_TX pin is disabled

Bit	Name	Reset	Access	Description
	1			The CAN_TX pin is enabled

41.5.21 CANn_MIRx_CMDMASK - Interface Command Mask 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																									0	0	0	0	0	0	0	0
Access																									RW	RW	RW	RW	RW	RW	RW	RW
Name																									WRRD	MASKACC	ARBACC	CONTROL	CLRINTPND	TXRQSTNEWDAT	DATAA	DATAB

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	WRRD	0	RW	**Write/Read RAM**

The control bits of the MIRx Command Mask Register specify the transfer direction and select which of the MIRx Message Buffer Registers are source or target of the data transfer. The other bits of MIRs Command Mask Register have different functions depending on the transfer direction.

Value	Mode	Description
0	READ	Transfer data from the Message Object addressed by the Command Request Register into the selected Message Buffer Registers.
1	WRITE	Transfer data from the selected Message Buffer Registers to the Message Object addressed by the Command Request Register.

6	MASKACC	0	RW	**Access Mask Bits**
5	ARBACC	0	RW	**Access Arbitration Bits**
4	CONTROL	0	RW	**Access Control Bits**
3	CLRINTPND	0	RW	**Clear Interrupt Pending Bit**

A read access to a Message Object can be combined with the clearing of the control bits INTPND and DATAVALID. The values of these bits transferred to the MIRx Message Control Register always reflect the status before clearing these bits.

2	TXRQSTNEWDAT	0	RW	**Transmission Request Bit/ New Data Bit**

If a transmission is requested by programming bit TXRQSTNEWDAT in the MIRx Command Mask Register, bit TXRQST in the MIRx Message Control Register will be ignored

1	DATAA	0	RW	**Access Data Bytes 0-3**
0	DATAB	0	RW	**CC Channel Mode**

These bits select the mode for Compare/Capture channel.

41.5.22 CANn_MIRx_MASK - Interface Mask 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	1	1		0x1FFFFFFF																												
Access	RW	RW		RW																												
Name	MXTD	MDIR		MASK																												

Bit	Name	Reset	Access	Description
31	MXTD	1	RW	**Mask Extended Identifier**
30	MDIR	1	RW	**Mask Message Direction**
29	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
28:0	MASK	0x1FFFFFFF	RW	**Identifier Mask**

41.5.23 CANn_MIRx_ARB - Interface Arbitration 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	0	0	0															0x00000000														
Access	RW	RW	RW															RW														
Name	MSGVAL	XTD	DIR															ID														

Bit	Name	Reset	Access	Description
31	MSGVAL	0	RW	**Message Valid**

The CPU must reset the MSGVAL bit of all unused Messages Objects during the initialization before it resets bit INIT in the CAN Control Register. This bit must also be reset before the identifier id[28:0], the control bits XTD, DIR, or the Data Length Code dlc[3:0] are modified, or if the Messages Object is no longer required.

Bit	Name	Reset	Access	Description
30	XTD	0	RW	**Extended Identifier**

Value	Mode	Description
0	STD	The 11-bit (standard) Identifier will be used for this Message Object.
1	EXT	The 29-bit (extended) Identifier will be used for this Message Object.

Bit	Name	Reset	Access	Description
29	DIR	0	RW	**Message Direction**

Value	Mode	Description
0	RX	On TXRQST, a Remote Frame with the identifier of this Message Object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this Message Object.
1	TX	On TXRQST, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TXRQST bit of this Message Object is set (if RMTEN = 1).

Bit	Name	Reset	Access	Description
28:0	ID	0x00000000	RW	**Message Identifier**

ID[28:0] is 29-bit Identifier for Extended Frame. ID[28:18] is 11-bit Identifier for Standard Frame. When 11-bit (standard) Identifiers are used for a Message Object, the identifiers of received Data Frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits MSK[28:18] are considered. The Arbitration Registers ID[28:0], XTD, and DIR are used to define the identifier and type of outgoing messages and are used (together with the mask registers MSK[28:0], MXTD, and MDIR) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and direction=receive (Data Frame) or direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with XTD=one, standard frames in Message Objects with XTD=zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

41.5.24 CANn_MIRx_CTRL - Interface Message Control Register

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																	0	0	0	0	0	0	0	0	0				0x0			
Access																	RWH	RWH	RW	RW	RW	RW	RW	RW	RW				RW			
Name																	DATAVALID	MESSAGEOF	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB				DLC			

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	DATAVALID	0	RWH	**New Data**
14	MESSAGEOF	0	RWH	**Message Lost (only Valid for Message Objects With Direction = Receive)**
13	INTPND	0	RW	**Interrupt Pending**
12	UMASK	0	RW	**Use Acceptance Mask** If the UMASK bit is set to one, the Message Object's mask bits have to be programmed during initialization of the Message Object before MSGVAL is set to one.
11	TXIE	0	RW	**Transmit Interrupt Enable**
10	RXIE	0	RW	**Receive Interrupt Enable**
9	RMTEN	0	RW	**Remote Enable**
8	TXRQST	0	RW	**Transmit Request**
7	EOB	0	RW	**End of Buffer** This bit is used to concatenate two or more Message Objects (up to 32) to build a FIFO Buffer. For single Message Objects (not belonging to a FIFO Buffer) this bit must always be set to one.
6:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3:0	DLC	0x0	RW	**Data Length Code** The Data Length Code of a Message Object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message. 0-8: Data Frame has 0-8 data bytes. 9-15: Data Frame has 8 data bytes.

41.5.25 CANn_MIRx_DATAL - Interface Data a 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	0x00								0x00								0x00								0x00							
Access	RW								RW								RW								RW							
Name	DATA3								DATA2								DATA1								DATA0							

Bit	Name	Reset	Access	Description
31:24	DATA3	0x00	RW	**Fourth Byte of CAN Data Frame**
23:16	DATA2	0x00	RW	**Third Byte of CAN Data Frame**
15:8	DATA1	0x00	RW	**Second Byte of CAN Data Frame**
7:0	DATA0	0x00	RW	**First Byte of CAN Data Frame**

41.5.26 CANn_MIRx_DATAH - Interface Data B 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	0x00								0x00								0x00								0x00							
Access	RW								RW								RW								RW							
Name	DATA7								DATA6								DATA5								DATA4							

Bit	Name	Reset	Access	Description
31:24	DATA7	0x00	RW	**Eight Byte of CAN Data Frame**
23:16	DATA6	0x00	RW	**Seventh Byte of CAN Data Frame**
15:8	DATA5	0x00	RW	**Sixth Byte of CAN Data Frame**
7:0	DATA4	0x00	RW	**Fifth Byte of CAN Data Frame**

41.5.27 CANn_MIRx_CMDREQ - Interface Command Request 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																	0										0x01					
Access																	R										RW					
Name																	BUSY										MSGNUM					

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	BUSY	0	R	**Busy Flag**

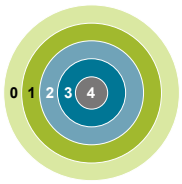
A message transfer is started as soon as the CPU has written the message number to the Command Request Register. With this write operation the busy bit is automatically set to '1' and signal CAN_WAIT_B is pulled LOW to notify the CPU that a transfer is in progress. After a wait time of 3 to 6 CAN_CLK periods, the transfer between the Interface Register and the Message RAM has completed. The busy bit is set back to zero and CAN_WAIT_B is set back to HIGH.

Value	Mode	Description
0	FALSE	Reset to zero when read/write action has finished.
1	TRUE	Set to one when writing to the MIRx Command Request Register.

14:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	MSGNUM	0x01	RW	**Message Number**

When a Message Number that is not valid is written into the Command Request Register, the Message Number will be transformed into a valid value and that Message Object will be transferred. There are 32 Message Objects in the Message RAM. To avoid conflicts between CPU accessing to the Message RAM and CAN message reception and transmission, the CPU cannot directly access the Message Objects, these accesses are handled via the MIRx Interface Registers. 1-32: Valid Message Number, the Message Object in the Message RAM is selected for data transfer. 0: Not a valid Message Number, interpreted as 32. 33-63: Not a valid Message Number, interpreted as 1-31.

42. 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.

42.1 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.

42.2 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 to 4 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

42.3 Functional Description

42.3.1 Overview

An overview of PDM is shown in Figure 42.1 PDM Overview on page 1815.

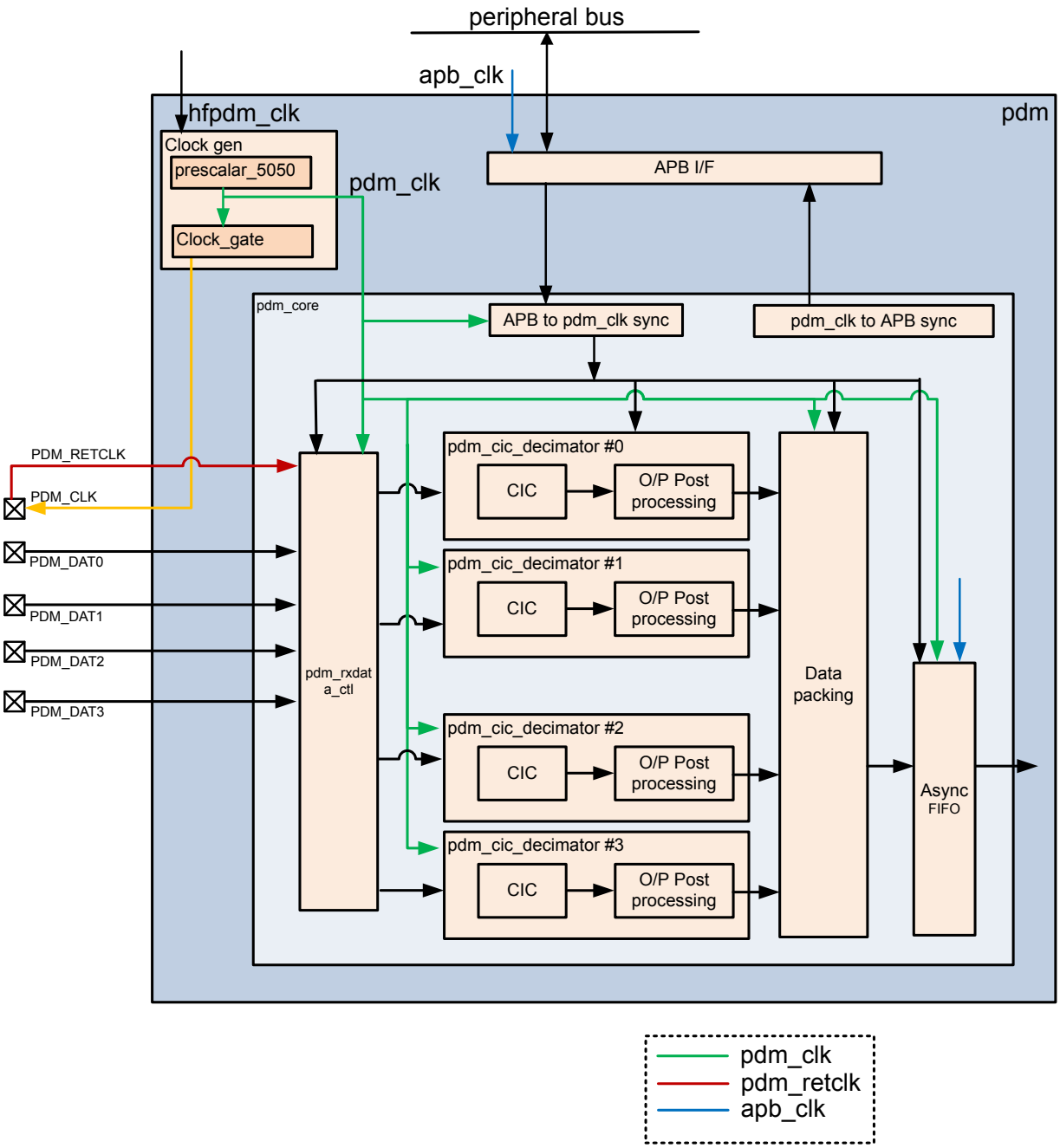


Figure 42.1. PDM Overview

42.3.2 PDM Clock Generation

The module generates a PDM output clock using an integer divider prescaler to divide down the module HFCLK_PDM. See 10.3.1.9 PDMCLK - PDM Core Clock 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.

42.3.3 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.

42.3.4 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 42.1. Maximum DSR vs Filter Order

N	Max DSR
3	1290
4	215
5	73

42.3.5 Multi Channel Operation

The PDM module has 4 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 to 4 channels.

In normal mode, there is one clock output and a separate 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.

The module also supports four channel stereo mode. This mode uses rising and falling clock edges to sample DAT0 for CH0 and CH1, and rising and falling clock edges to sample DAT3 for CH3 and CH4.

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.

42.3.6 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.

42.3.7 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 is set to 2 (three channels) and the DATAFORMAT field is set to DOUBLE16, then the FIFO is written in the following 32-bit packed format:

Table 42.2. Data packing for DOUBLE16 with 3 channels

UPPER16BIT	LOWER16BIT
CH1	CH0
CH0	CH2
CH2	CH1

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.

42.3.8 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.

42.3.9 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.

42.3.10 PDM Energy Modes

- The PDM module function in EM0/EM1 mode.
- All the PDM registers are retained in EM2; after waking from EM2 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 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.
  - Upon EM2 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.

42.3.11 Debug Mode

When CPU is halted in debug mode, the PDM will continue to run and DMA will capture the data from the FIFO.

42.3.12 Pin Configurations

A common GPIO pin is used as the clock output for all four sensors/microphones and up to 4x DATA ports.

Five fixed route locations are available and available pin locations are detailed in the device data sheet. Pin locations can be selected using the PDMn_ROUTELOC0 / PDMn_ROUTELOC1 registers. Pins are enabled using the PDMn_ROUTEPEN register. Set the slew rate for the CLK pin to 7.

Pin configurations are shown below.

Table 42.3. Pin Configurations

Signal	Route enable	Route location	I/O	GPIO MODEn
PDM_CLK	PDM_ROUTEPEN.CLKPEN	PDM_ROUTELOC1.CLKLOC	O	Pushpull/Pushpullalt
PDM_DAT0	PDM_ROUTEPEN.DAT0PEN	PDM_ROUTELOC0.DAT0LOC	I	Input
PDM_DAT1	PDM_ROUTEPEN.DAT1PEN	PDM_ROUTELOC0.DAT1LOC	I	Input
PDM_DAT2	PDM_ROUTEPEN.DAT2PEN	PDM_ROUTELOC0.DAT2LOC	I	Input
PDM_DAT3	PDM_ROUTEPEN.DAT3PEN	PDM_ROUTELOC0.DAT3LOC	I	Input

42.3.13 External Device Interface

Based on the data mux architecture in the PDM front end RXDATA control module, external device connections to the PDM interface must follow specific rules.

- When connecting two mono microphones or two sensors, the devices must be connected to PDM_DAT0 and PDM_DAT1.

  For three mono devices, connect to PDM_DAT0, PDM_DAT1, and PDM_DAT2.

  For four mono devices, connect to PDM_DAT0, PDM_DAT1, PDM_DAT2, and PDM_DAT3.
- When connecting two stereo microphones, the first set of stereo microphones must be connected to PDM_DAT0 and the second pair to PDM_DAT2.
- When connecting one stereo microphone and one mono device, the stereo microphone must be connected to PDM_DAT0 and the mono device to PDM_DAT2.

  When connecting one stereo microphone and two mono devices, two configurations are possible:
  - Connect the stereo device to PDM_DAT0 and the two mono devices to PDM_DAT2 and PDM_DAT3.
  - Connect the mono devices to PDM_DAT0 and PDM_DAT1, and the stereo device to PDM_DAT2.

42.3.14 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 START/STOP/FIFOFL/CLEAR commands are issued, poll for the corresponding syncbusy bits (STARTBUSY/STOPBUSY/FIFOFLBUSY) in the SYNCBUSY register.

  Similarly when the DSR/GAIN fields in CTRL register are configured, poll for CTRLBUSY bit in the SYNCBUSY register.
- The START command is common for all channels of the Filter. When NUMCH is set to 3, i.e. when four channels are configured, all four 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 3 (four channels), then Minimum DSR should be 7.
- If the DATAFORMAT selected in PDMn_CFG0 is DOUBLE16, then Minimum DSR > ((NUMCH + 1)/2 + 2).

  Example: If NUMCH is 3 (four channels) and the DATAFORMAT is DOUBLE16, then Minimum DSR should be 5.

42.3.14.1 Configuration example 1

Sample program for configuring PDM

- Set 1p2v or 1p0v mode
- Enable clocks in CMU, configure PDM clock in CMU

```
/* Configure GPIO clock, LDMA clock */
CMU->HFBUSCKEN0 = CMU->HFBUSCLKEN0 | CMU_HFBUSCLKEN0_GPIO | CMU_HFBUSCLKEN0_LDMA
/* Enable PCLK of PDM peripheral */
CMU->HFPERCLKEN0 = (CMU->HFPERCLKEN0 | CMU_HFPERCLKEN0_PDM)
/* If required choose PER clock prescale value */
CMU->HFPRESC = (prescale_value << _CMU_HFPRESC_PRESC_SHIFT)
```

Select PDM clock AND Enable PDM core clock

```
CMU->PDMCTRL = CMU_PDMCTRL_PDMCLK_HFRCO | CMU_PDMCTRL_PDMCLKEN;
```

- Configure GPIOs using ROUTEPEN/ROUTELOC0/ROUTELOC1 registers if PDM

  Set CLKPIN 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

42.4 Applications

42.4.1 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 1.576 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	-

42.4.2 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		

42.5 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	PDM Module enable Register
0x008	PDM_CTRL	RW	PDM Core Control Register
0x00C	PDM_CMD	W1	PDM Core Command Register
0x010	PDM_STATUS	R	PDM Status register
0x014	PDM_CFG0	RW	PDM Core Configuration Register0
0x018	PDM_CFG1	RW	PDM Core Configuration Register1
0x020	PDM_RXDATA	R(a)	PDM Received Data Register
0x040	PDM_IF	R	Interrupt Flag Register
0x044	PDM_IFS	W1	Interrupt Flag Set Register
0x048	PDM_IFC	(R)W1	Interrupt Flag Clear Register
0x04C	PDM_IEN	RW	Interrupt Enable Register
0x050	PDM_ROUTEPEN	RW	I/O LOCATION Enable Register
0x054	PDM_ROUTELOC0	RW	I/O LOCATION Register
0x058	PDM_ROUTELOC1	RW	I/O LOCATION Register
0x060	PDM_SYNCBUSY	R	Synchronization Busy Register

42.6 Register Description

42.6.1 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	0x00000000																															
Access	R																															
Name	IPVERSION																															

Bit	Name	Reset	Access	Description
31:0	IPVERSION	0x00000000	R	**IP VERSION**
	Indicates the version of IP			

42.6.2 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																																0
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	0	RW	**Module enable**
	when set to 1, module is enabled			

Value	Mode	Description
0	DISABLE	Disable module
1	ENABLE	Enable module

42.6.3 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	0												0x000															0x00				
Access	RW												RW															RW				
Name	OUTCLKEN												DSR															GAIN				

Bit	Name	Reset	Access	Description
31	OUTCLKEN	0	RW	**PDM Clock enable**
	To be enabled when in EM01.			
30:20	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
19:8	DSR	0x000	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	0x00	RW	**Selects Gain factor of DCF**
	Selects Gain factor of DCF .			

42.6.4 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																0								0				0				0
Access																W1								W1				W1				W1
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	0	W1	**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	0	W1	**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	0	W1	**Stop DCF**
	One hot for selectively stopping the filter.			
3:1	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
0	START	0	W1	**Start DCF**
	One hot for selectively starting the filter.			

42.6.5 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					1	0				0
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 PDM FIFO current empty slots.			
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5	EMPTY	1	R	**FIFO EMPTY Status**
	Indicates PDM FIFO is Empty.			
4	FULL	0	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	0	R	**PDM is active**
	Indicates PDM is running.			

42.6.6 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					0	0	0	0							0	0			0x0			0x0					0x0				0x0	
Access					RW	RW	RW	RW							RW	RW			RW			RW					RW				RW	
Name					CH3CLKPOL	CH2CLKPOL	CH1CLKPOL	CH0CLKPOL							STEREOMODECH23	STEREOMODECH01			FIFODVL			DATAFORMAT					NUMCH				FORDER	

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	CH3CLKPOL	0	RW	**CH3 CLK Polarity**
	Configure CH3 CLK Polarity.			
	Value	Mode		Description
	0	NORMAL		Input data clocked on rising clock edge.
	1	INVERT		Input data clocked on falling clock edge.
26	CH2CLKPOL	0	RW	**CH2 CLK Polarity**
	Configure CH2 CLK Polarity.			
	Value	Mode		Description
	0	NORMAL		Input data clocked on rising clock edge.
	1	INVERT		Input data clocked on falling clock edge.
25	CH1CLKPOL	0	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	0	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.
23:18	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		

Bit	Name	Reset	Access	Description
17	STEREOMODECH23	0	RW	**Stereo mode CH23**

Stereo mode Configuration for Channel pair CH2 and CH3.

Value	Mode	Description
0	DISABLE	No Stereo mode.
1	CH23ENABLE	CH2 and CH3 in Stereo mode.

16	STEREOMODECH01	0	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.
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.

7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*

5:4	NUMCH	0x0	RW	**Number of Channels**

Number of Channels.

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	0	ONE		Only one Channel.
	1	TWO		Two Channels.
	2	THREE		Three Channels.
	3	FOUR		Four 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.

42.6.7 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																							0x000									
Access																							RW									
Name																							PRESC									

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	PRESC	0x000	RW	**Prescalar Setting for PDM sample**
	Generate Decimation filter clock			

42.6.8 PDM_RXDATA - PDM Received Data Register (Actionable Reads)

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	0x00000000																															
Access	R																															
Name	RXDATA																															

Bit	Name	Reset	Access	Description
31:0	RXDATA	0x00000000	R	**PDM received data**
	Use this register to access data from FIFO.			

42.6.9 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																													0	0	0	0
Access																													R	R	R	R
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	0	R	**FIFO Undeflow Interrupt Flag**
	Set when a FIFO Underflow condition occurs.			
2	OF	0	R	**FIFO Overflow Interrupt Flag**
	Set when a FIFO Overflow condition occurs.			
1	DVL	0	R	**Data Valid Level Interrupt Flag**
	Set when the FIFO reaches the watermark level.			
0	DV	0	R	**Data Valid Interrupt Flag**
	This interrupt is set after valid data available in FIFO.			

42.6.10 PDM_IFS - Interrupt Flag Set 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																													0	0	0	0
Access																													W1	W1	W1	W1
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	0	W1	**Set UF Interrupt Flag**
	Write 1 to set the UF interrupt flag			
2	OF	0	W1	**Set OF Interrupt Flag**
	Write 1 to set the OF interrupt flag			
1	DVL	0	W1	**Set DVL Interrupt Flag**
	Write 1 to set the DVL interrupt flag			
0	DV	0	W1	**Set DV Interrupt Flag**
	Write 1 to set the DV interrupt flag			

42.6.11 PDM_IFC - Interrupt Flag Clear 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																													0	0	0	0
Access																													(R)W1	(R)W1	(R)W1	(R)W1
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	0	(R)W1	**Clear UF Interrupt Flag**
	Write 1 to clear the UF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
2	OF	0	(R)W1	**Clear OF Interrupt Flag**
	Write 1 to clear the OF interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
1	DVL	0	(R)W1	**Clear DVL Interrupt Flag**
	Write 1 to clear the DVL interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			
0	DV	0	(R)W1	**Clear DV Interrupt Flag**
	Write 1 to clear the DV interrupt flag. Reading returns the value of the IF and clears the corresponding interrupt flags (This feature must be enabled globally in MSC.).			

42.6.12 PDM_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																													0	0	0	0
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	0	RW	**UF Interrupt Enable**
	Enable/disable the UF interrupt			
2	OF	0	RW	**OF Interrupt Enable**
	Enable/disable the OF interrupt			
1	DVL	0	RW	**DVL Interrupt Enable**
	Enable/disable the DVL interrupt			
0	DV	0	RW	**DV Interrupt Enable**
	Enable/disable the DV interrupt			

42.6.13 PDM_ROUTEPEN - I/O LOCATION Enable 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																								0					0	0	0	0
Access																								RW					RW	RW	RW	RW
Name																								CLKPEN					DAT3PEN	DAT2PEN	DAT1PEN	DAT0PEN

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	CLKPEN	0	RW	**CLK I/O Enable**
	When enabled connects input of CLK0 to pin.			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	DAT3PEN	0	RW	**DAT3 I/O Enable**
	When enabled connects input of DAT3 to pin.			
2	DAT2PEN	0	RW	**DAT2 I/O Enable**
	When enabled connects input of DAT2 to pin.			
1	DAT1PEN	0	RW	**DAT1 I/O Enable**
	When enabled connects input of DAT1 to pin.			
0	DAT0PEN	0	RW	**DAT0 I/O Enable**
	When enabled connects input of DAT0 to pin.			

42.6.14 PDM_ROUTELOC0 - I/O LOCATION 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			0x00								0x00								0x00								0x00					
Access			RW								RW								RW								RW					
Name			DAT3LOC								DAT2LOC								DAT1LOC								DAT0LOC					

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	DAT3LOC	0x00	RW	**I/O Location for DAT3 pins**

Selects location of DAT3 pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4

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*		
21:16	DAT2LOC	0x00	RW	**I/O Location for DAT2 pins**

Selects location of DAT2 pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4

Bit	Name	Reset	Access	Description
15:14	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
13:8	DAT1LOC	0x00	RW	**I/O Location for DAT1 pins**

Selects location of DAT1 pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

Bit	Name	Reset	Access	Description
	3	LOC3		Location 3
	4	LOC4		Location 4
7:6	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
5:0	DAT0LOC	0x00	RW	**I/O Location for DAT0 pins**
	Selects location of DAT0 pins.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4

42.6.15 PDM_ROUTELOC1 - I/O LOCATION 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																											0x00				
Access																											RW				
Name																											CLKLOC				

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	CLKLOC	0x00	RW	**I/O Location for CLK pin**
	Selects location of CLK0 pins.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4

42.6.16 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																						0		0					0	0	0	0
Access																						R		R					R	R	R	R
Name																						CTRLREGBUSY		PRESCBUSY					FIFOFLBUSY	CLEARBUSY	STOPBUSY	STARTBUSY

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	CTRLREGBUSY	0	R	**CTRLREGBUSY busy**
	Indicates CTRL sync busy.			
9	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
8	PRESCBUSY	0	R	**PRESC Sync busy**
	Indicates PRESC sync busy.			
7:4	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in 1.2 Conventions*		
3	FIFOFLBUSY	0	R	**FIFO Flush Sync busy**
	Indicates FIFO Flush sync busy.			
2	CLEARBUSY	0	R	**CLEAR sync busy**
	Indicates CLEAR sync busy.			
1	STOPBUSY	0	R	**STOP sync busy**
	Indicates STOP sync busy.			
0	STARTBUSY	0	R	**START sync busy**
	Indicates START sync busy.			

43. Revision History

Revision 1.1

February, 2022

- Description of the Bank Switch Disable bit in Configuration Lock Word 1 improved in 6.3.2 Lock Bits (LB) Page Description.
- Note added to 6.3.2 Lock Bits (LB) Page Description explaining that a hard reset is required for changes to the lock bits to take effect.
- Note added to 6.3.12 Bank Switching Operation explaining that CLW1 bit 0 must previously be programmed to 0 to enable bank switching.
- 6.3.13.1 Read-While-Write updated to show the mapping of flash pages allocated to the lock bits, user data, and bootloader regions and their association with the two physical flash instances.

Revision 1.0

March, 2021

- Added description of VLP and ACMP startup behavior when External Override Interface is enabled.
- Corrected Information Block address in Table 6.1 MSC Flash Memory Mapping on page 169.
- Updated the Note under 22.3.1.6 Underflow/Overflow From Neighboring Timer.
- Added a Note to 28.3.3.2 Scan Mode.
- Updated 28.3.10.9 Temperature Measurement and added an additional Note.
- Made minor changes and fixed typos throughout the document.

Revision 0.5

December, 2018

- 4.6 DI Page Entry Map: Added details for future module support.
- Added section for 20. UART - Universal Asynchronous Receiver/ Transmitter.
- 38. USB - Universal Serial Bus Controller: Improved register descriptions.
- 39. SDIO - SDIO Host controller: Removed specific performance numbers (documented in datasheet specification tables).
- 42. PDM - PDM Interface: Additional functional details added.

Revision 0.1

June, 2018

Initial release.

Appendix 1. Abbreviations

This section lists abbreviations used in this document.

Table 1.1. Abbreviations

Abbreviation	Description
ACMP	Analog Comparator
ADC	Analog to Digital Converter
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".
ALE	Address Latch Enable
AUXHFRCO	Auxiliary High Frequency RC Oscillator.
CC	Compare / Capture
CIC	Cascaded Integrator Comb
CLK	Clock
CMD	Command
CMU	Clock Management Unit
CTRL	Control
DAC	Digital to Analog Converter
DBG	Debug
DMA	Direct Memory Access
DRD	Dual Role Device
DTI	Dead Time Insertion
EBI	External Bus Interface
EFM	Energy Friendly Microcontroller
EM	Energy Mode
EM0 Active	Energy Mode 0 (also called active mode)
EM1 Sleep to EM4 Hibernate/Shutoff	Energy Mode 1 to Energy Mode 4 (also called low energy modes)
EMU	Energy Management Unit
ENOB	Effective Number of Bits
FS	Full-speed
GPIO	General Purpose Input / Output
HFRCO	High Frequency RC Oscillator
HFXO	High Frequency Crystal Oscillator
HW	Hardware
I^2C	Inter-Integrated Circuit interface
LCD	Liquid Crystal Display
LESENSE	Low Energy Sensor Interface

Abbreviation	Description
LETIMER	Low Energy Timer
LEUART	Low Energy Universal Asynchronous Receiver Transmitter
LFRCO	Low Frequency RC Oscillator
LFXO	Low Frequency Crystal Oscillator
LS	Low-speed
MAC	Media Access Controller
NVIC	Nested Vector Interrupt Controller
OSR	Oversampling Ratio
OTG	On-the-go
PCNT	Pulse Counter
PCM	Pulse Code Modulation
PDM	Pulse Density Modulation
PHY	Physical Layer
PRS	Peripheral Reflex System
PWM	Pulse Width Modulation
RC	Resistance and Capacitance
RMU	Reset Management Unit
RTC	Real Time Clock
SAR	Successive Approximation Register
SOF	Start of Frame
SPI	Serial Peripheral Interface
SW	Software
TRNG	True Random Number Generator
UART	Universal Asynchronous Receiver Transmitter
USART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VMON	Voltage supply monitor
WDOG	Watchdog timer
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

www.silabs.com