XS1-XAU8A-10-FB265 Datasheet

2015/04/28 Document Number: X005109,

Table of Contents

1 xCORE eXtended Architecture

The XS1-XAU Series is a programmable SOC that combines the flexibility and determinism of xCORE multicore microcontrollers with the low energy use and fixed interfaces of an ARM core to deliver:

- Programmable system-on-chip integration
- Multicore performance for demanding applications
- Extremely low energy for long battery life
- Easy to use - completely programmable in C

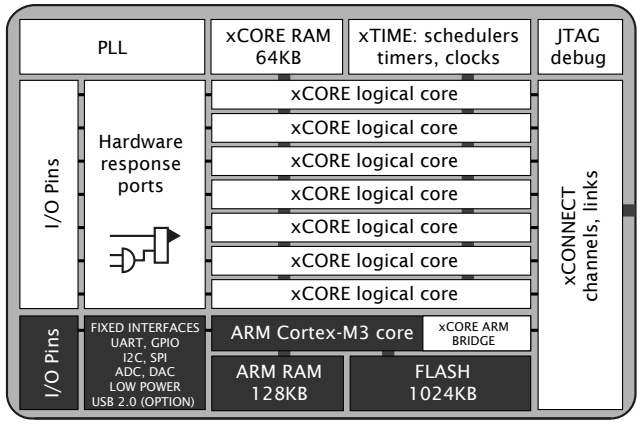


Figure 1: XS1-XAU: 8 core device

Key features of XS1-XAU devices include:

- **Logical cores** Each device has an xCORE Tile that contains eight independent 32-bit xCORE logical processors, which can execute tasks such as computational code, advanced DSP code, control software (including logic decisions and executing a state machine) or software that handles I/O. One of the cores is dedicated to a communication bridge with the ARM-core. Section 5.2

  The logical cores are triggered by events generated by hardware resources such as the I/O pins, other cores and timers. Once triggered, a core runs independently and concurrently to other cores. An active core runs to completion once triggered or can be paused by other events. Section 5.2

- **xTIME scheduler** The xTIME scheduler performs functions similar to an RTOS, in hardware on the xCORE Tile. It services and synchronizes all events generated by the Tile resources, with tasks in a single context so there is no requirement for interrupt handler routines. Section 5.3

- **Channels and channel ends** Tasks running on logical cores communicate using channels formed between two channel ends. Data can be passed synchronously or asynchronously between the channel ends assigned to the communicating tasks. Section 5.6

- **xCONNECT Switch** Logical cores on a tile can communicate with cores on other tiles over a high performance network of links that are routed through a hardware xCONNECT Switch. Section 5.7
- **Ports** The xCORE I/O pins are connected to the processing cores by Hardware Response ports. The port logic can drive its pins high and low, or it can sample the value on its pins optionally waiting for a particular condition. Section 5.4
- **Counters** Each xCORE port has a counter that can be used to control the time at which data is input or output. The counter value can be used to provide precise control of response times. See Section 5.4
- **Clock blocks** xCORE devices include a set of programmable clock blocks that can be used to govern the rate at which ports execute. Each xCORE tile has a reference clock block that runs at 100 MHz and additional clock blocks that can be configured to run at different speeds. Section 5.5
- **Timers** The xCORE Tile has ten 32-bit counters called timers that run relative to the xCORE Tile Reference clock. Section 5.8
- **xCORE Memory** Each xCORE Tile integrates a bank of SRAM for instructions and data, and a block of one-time programmable (OTP) memory that can be configured for system wide security features. Section 9
- **PLL** The PLL is used to create a high-speed clock that defines the frequency of the xCORE reference clock, xCONNECT switch and xCORE Tile clock. Section 7
- **ARM CORTEX-M3** ARM Cortex-M3 32-bit RISC processor with a Memory Protection Unit and a Wake-up Interrupt Controller to handle interrupts triggered while the CPU is asleep. Section 5.9
- **xCORE-ARM Bridge** The xCORE Tile and ARM-core communicate with each other via the xCORE-ARM bridge, using the XAB library that is available with the development tools. Section 6
- **Fixed interfaces/peripherals** The ARM-core includes modules and fixed peripherals that are driven by the Cortex-M3. Section 5.10
- **ARM Memory** ARM-core each has an SRAM module for storing application code and data. It also has a block of integrated flash memory for storing program code, user data and flash lock bits. Section 9
- **ARM GPIO** The ARM GPIO pins are bonded out and available for general input or output, and peripheral configuration. All pins are blinded out but some are not available when the xCORE-ARM bridge is active. Section 5.11
- **JTAG** The xCORE Tile and ARM-core have separate JTAG modules that can be used for loading programs, boundary scan testing, incircuit source-level debugging and programming the memory. Section 10

1.1 Software

The xCORE tiles are programmed using C, C++ or xC (C with multicore extensions). XMOS provides tested and proven software libraries, which allow you to quickly configure the hardware.

For information on accessing the ARM digital and analog I/O, or peripherals in your applications, we recommend that you refer to the application notes on the Silicon Labs website.

1.2 xTIMEcomposer Studio

The xTIMEcomposer tools support application development for xCORE-XA devices and boards. Extensions have been made to xTIMEcomposer to support multi-architecture projects; the tools now include a full command line toolchain for supporting the ARM Cortex-M series as well as the xCORE Tile. This and the additional support added to the build infrastructure for XMOS applications allows single project development of applications within the xTIMEcomposer design environment.

xTIMEcomposer can be driven from either a graphical development environment, or the command line. The tools are supported on Windows, Linux and MacOS X and available at no cost from xmos.com/downloads. Information on using the tools is provided in the xTIMEcomposer User Guide, X3766.

2 XS1-XAU8A-10-FB265 Features

- **Eight-Core Multicore Microcontroller with Advanced Multi-Core RISC Architecture**
  - **8 xCORE real-time 32-bit logical cores** that share up to 500 MIPS
    - One core dedicated to xCORE-ARM Bridge
  - **ARM Cortex-M3** 32-bit processor running up to 48 MHz
- **xCORE resources**
  - Each logical core has:
    - Guaranteed throughput of between 1/4 and 1/8 of tile MIPS
    - 16x32bit dedicated registers
  - 159 high-density 16/32-bit instructions
    - All have single clock-cycle execution (except for divide)
    - 32x32→64-bit MAC instructions for DSP, arithmetic and user-definable cryptographic functions
  - **40 general-purpose I/O pins**
    - Configurable as input or output
    - Up to 14 x 1bit port, 4 x 4bit port, 3 x 8bit port, 1 x 16bit port
    - 3 xCONNECT links
  - Port sampling rates of up to 60 MHz with respect to an external clock
  - 32 channel ends for communication with other cores, on or off-chip
  - **Hardware resources**
    - 6 clock blocks
    - 10 timers
  - **Memory**
    - 64KB internal single-cycle SRAM for code and data storage
    - 8KB internal OTP for application boot code
  - **xCORE JTAG Module for On-Chip Debug**
  - **xCORE Security Features**
    - Programming lock disables debug and prevents read-back of memory contents
    - AES bootloader ensures secrecy of IP held on external flash memory
- **ARM core**
  - **Fixed peripherals - ARM core**
    - Serial: UART, USART, LEUART, I2C, USB
    - Analog: ADC, DAC, OPAMP, Pulse counter
  - **70 general-purpose I/O pins**
    - Configurable as input or output
  - **Energy management**
    - Low energy modes down to 100nA
  - **Memory**
    - 128KB internal single-cycle SRAM for code and data storage
    - 1024KB SPI FLASH
- **Ambient Temperature Range**
  - Commercial qualification: 0°C to 70°C
  - Industrial qualification: -40°C to 85°C
- **xCORE Speed Grade**
  - 5: 500 MIPS
- **265-pin FBGA package 0.8 mm pitch**

3 Pin Configuration

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	GND	VDDIO	VDDIO	4C X0D12	4D X0D14	4E X0D16	4G X0D18	4F X0D20	4H X0D22	5H X0D36	5G X0D38	4E NC	4F NC	4F NC	4E NC	PB15	NC	VDDIO	GND
B	VDDCORE	VDDCORE	VDDIO	4C X0D13	4D X0D15	4E X0D17	4G X0D19	4F X0D21	4H X0D23	5H X0D37	5G X0D39	4E NC	4F NC	4F NC	4E NC	PD13	VDDIO	VDDCORE	VDDCORE
C	VDDCORE	VDDCORE	VDDCORE	VDDIO	VDDIO	6D X0D40	6D X0D41	6D X0D43	NC	NC	NC	NC	NC	NC	VDDIO	VDDIO	VDDCORE	5J NC	PD10
D	1C X0D10	1D X0D11	VDDCORE	VDDCORE	VDDIO	NC	8D X0D42	VDDCORE	VDDCORE	GND	GND	NC	NC	NC	VDDIO	VDDCORE	VDDCORE	1I NC	32A X0D50
E	4A X0D08	4A X0D09	32A X0D52	VDDCORE												NC	NC	32A X0D49	PF7
F	1B X0D06	1B X0D07	32A X0D53	GND												NC	NC	PF6	PF5
G	RST_N	CLK	32A X0D54	GND												NC	NC	PF12	PF2
H	4B X0D04	4B X0D05	32A X0D55	VDDCORE				GND	GND	GND	GND	GND				NC	PF4	USB VBUS	PF3
J	4A X0D02	4A X0D03	MODE[0]	VDDCORE				GND	GND	GND	GND	GND				NC	NC	PF1	PF0
K	1A X0D00	1B X0D01	MODE[1]	NC				GND	GND	GND	GND	GND				NC	USB VREGD	PF10	PF11
L	PLLVSS	DEBUG_N	PD14	NC				GND	GND	GND	GND	GND				NC	USB VREGI	PC14	PC15
M	PLLVDD	TMS	MODE[2]	NC				GND	GND	GND	GND	AVSS				PE4	PE7	PC12	PC13
N	NC	TDI	MODE[3]	NC												PE5	PE6	PC10	PC11
P	TCK	1K X0D34	32A NC	NC												PE2	PE1	PC8	PC9
R	TDO	32A NC	32A NC	NC												AVSS	PE0	PE3	DECOUPLE
T	1 X0D35	32A NC	NC	VDDEM	VDDEM	NC	NC	NC	PA11	PA12	PA13	PB9	PB10	AVDD	VDDEM	VDDEM	PD8	VDD_DREG	PC7
U	32A NC	32A NC	VDDEM	VDDEM	NC	PD15	PB4	PB5	PB6	PA7	PA8	PA9	PA10	PB13	AVDD	VDDEM	VDDEM	PD7	PC6
V	VDDEM	VDDEM	VDDEM	32A NC	PB0	PB2	PC1	PC2	PC4	PB8	RESETn	PB11	NC	PD1	PD3	AVDD	VDDEM	PD5	PD6
W	GND	VDDEM	VDDEM	32A NC	PB1	PB3	PC0	PC5	PC3	PB7	PA14	PB12	PD0	PD2	PD4	AVDD	VDDEM	VDDEM	GND

4 Signal Description

Module	Signal	Function	Type	Active	Properties
	PU=Pull Up, PD=Pull Down, ST=Schmitt Trigger Input, OT=Output Tristate, S=Switchable R_S=Required for SPI boot (§8)				
Power	PLLVDD	Analog PLL power	PWR	—	
	PLLVSS	Analog ground for PLL	GND	—	
	VDDCORE	Core voltage supply	PWR	—	
	VDDIO	Digital I/O power	PWR	—	
Clocks	CLK	PLL reference clock	Input	—	PD, ST
	MODE[3:0]	Boot mode select	Input	—	PU, ST
JTAG	DEBUG_N	Multi-chip debug	I/O	Low	PU
	RST_N	Global reset input	Input	Low	PU, ST
	TCK	Test clock	Input	—	PU, ST
	TDI	Test data input	Input	—	PU, ST
	TDO	Test data output	Output	—	PD, OT
	TMS	Test mode select	Input	—	PU, ST
I/O	X0D00	$P1A^0$	I/O	—	PD_S, R_S
	X0D01	XLA^4_{out} $P1B^0$	I/O	—	PD_S, R_S
	X0D02	XLA^3_{out} $P4A^0$ $P8A^0$ $P16A^0$ $P32A^{20}$	I/O	—	PD_S
	X0D03	XLA^2_{out} $P4A^1$ $P8A^1$ $P16A^1$ $P32A^{21}$	I/O	—	PD_S
	X0D04	XLA^1_{out} $P4B^0$ $P8A^2$ $P16A^2$ $P32A^{22}$	I/O	—	PD_S
	X0D05	XLA^0_{out} $P4B^1$ $P8A^3$ $P16A^3$ $P32A^{23}$	I/O	—	PD_S
	X0D06	XLA^0_{in} $P4B^2$ $P8A^4$ $P16A^4$ $P32A^{24}$	I/O	—	PD_S
	X0D07	XLA^1_{in} $P4B^3$ $P8A^5$ $P16A^5$ $P32A^{25}$	I/O	—	PD_S
	X0D08	XLA^2_{in} $P4A^2$ $P8A^6$ $P16A^6$ $P32A^{26}$	I/O	—	PD_S
	X0D09	XLA^3_{in} $P4A^3$ $P8A^7$ $P16A^7$ $P32A^{27}$	I/O	—	PD_S
	X0D10	XLA^4_{in} $P1C^0$	I/O	—	PD_S, R_S
	X0D11	$P1D^0$	I/O	—	PD_S, R_S
	X0D12	$P1E^0$	I/O	—	PD_S
	X0D13	XLB^4_{out} $P1F^0$	I/O	—	PD_S
	X0D14	XLB^3_{out} $P4C^0$ $P8B^0$ $P16A^8$ $P32A^{28}$	I/O	—	PD_S
	X0D15	XLB^2_{out} $P4C^1$ $P8B^1$ $P16A^9$ $P32A^{29}$	I/O	—	PD_S
	X0D16	XLB^1_{out} $P4D^0$ $P8B^2$ $P16A^{10}$	I/O	—	PD_S
	X0D17	XLB^0_{out} $P4D^1$ $P8B^3$ $P16A^{11}$	I/O	—	PD_S
	X0D18	XLB^0_{in} $P4D^2$ $P8B^4$ $P16A^{12}$	I/O	—	PD_S
	X0D19	XLB^1_{in} $P4D^3$ $P8B^5$ $P16A^{13}$	I/O	—	PD_S
	X0D20	XLB^2_{in} $P4C^2$ $P8B^6$ $P16A^{14}$ $P32A^{30}$	I/O	—	PD_S
	X0D21	XLB^3_{in} $P4C^3$ $P8B^7$ $P16A^{15}$ $P32A^{31}$	I/O	—	PD_S
	X0D22	XLB^4_{in} $P1G^0$	I/O	—	PD_S
	X0D23	$P1H^0$	I/O	—	PD_S
	X0D34	$P1K^0$	I/O	—	PD_S
	X0D35	$P1L^0$	I/O	—	PD_S

(continued)

Module	Name	Function	Type	Active	Properties
I/O	X0D36	P1M[0] P8D[0] P16B[8]	I/O	—	PD_S
	X0D37	P1N[0] P8D[1] P16B[9]	I/O	—	PD_S
	X0D38	P1O[0] P8D[2] P16B[10]	I/O	—	PD_S
	X0D39	P1P[0] P8D[3] P16B[11]	I/O	—	PD_S
	X0D40	P8D[4] P16B[12]	I/O	—	PD_S
	X0D41	P8D[5] P16B[13]	I/O	—	PD_S
	X0D42	P8D[6] P16B[14]	I/O	—	PD_S
	X0D43	P8D[7] P16B[15]	I/O	—	PU_S
	X0D49	P32A[0]	I/O	—	PD_S
	X0D50	P32A[1]	I/O	—	PD_S
	X0D52	XLC^1_{out} P32A[3]	I/O	—	PD_S
	X0D53	XLC^0_{out} P32A[4]	I/O	—	PD_S
	X0D54	XLC^0_{in} P32A[5]	I/O	—	PD_S
	X0D55	XLC^1_{in} P32A[6]	I/O	—	PD_S
ARM Power	AVDD	Supply and reference voltage	power	—	
	AVSS	Digital ground	power	—	
	GND	Digital ground	GND	—	
	VDDEM	Power	power	—	
	VDD_DREG	Power	power	—	
ARM Output	USB_VREGO	USB power out	output	—	
ARM I/O	DECOUPLE	Power decouple	I/O	—	
	PA10	General-purpose I/O	I/O	—	
	PA11	General-purpose I/O	I/O	—	
	PA12	General-purpose I/O	I/O	—	
	PA13	General-purpose I/O	I/O	—	
	PA14	General-purpose I/O	I/O	—	
	PA7	General-purpose I/O	I/O	—	
	PA8	General-purpose I/O	I/O	—	
	PA9	General-purpose I/O	I/O	—	
	PB0	General-purpose I/O	I/O	—	
	PB1	General-purpose I/O	I/O	—	
	PB10	General-purpose I/O	I/O	—	
	PB11	General-purpose I/O	I/O	—	
	PB12	General-purpose I/O	I/O	—	
	PB13	General-purpose I/O	I/O	—	
	PB15	General-purpose I/O	I/O	—	
	PB2	General-purpose I/O	I/O	—	
	PB3	General-purpose I/O	I/O	—	
	PB4	General-purpose I/O	I/O	—	
	PB5	General-purpose I/O	I/O	—	
	PB6	General-purpose I/O	I/O	—	
	PB7	General-purpose I/O	I/O	—	
	PB8	General-purpose I/O	I/O	—	

(continued)

Module	Name	Function	Type	Active	Properties
ARM I/O	PB9	General-purpose I/O	I/O	—	
	PC0	General-purpose I/O	I/O	—	
	PC1	General-purpose I/O	I/O	—	
	PC10	General-purpose I/O	I/O	—	
	PC11	General-purpose I/O	I/O	—	
	PC12	General-purpose I/O	I/O	—	
	PC13	General-purpose I/O	I/O	—	
	PC14	General-purpose I/O	I/O	—	
	PC15	General-purpose I/O	I/O	—	
	PC2	General-purpose I/O	I/O	—	
	PC3	General-purpose I/O	I/O	—	
	PC4	General-purpose I/O	I/O	—	
	PC5	General-purpose I/O	I/O	—	
	PC6	General-purpose I/O	I/O	—	
	PC7	General-purpose I/O	I/O	—	
	PC8	General-purpose I/O	I/O	—	
	PC9	General-purpose I/O	I/O	—	
	PD0	General-purpose I/O	I/O	—	
	PD1	General-purpose I/O	I/O	—	
	PD10	General-purpose I/O	I/O	—	
	PD13	General-purpose I/O	I/O	—	
	PD14	General-purpose I/O	I/O	—	
	PD15	General-purpose I/O	I/O	—	
	PD2	General-purpose I/O	I/O	—	
	PD3	General-purpose I/O	I/O	—	
	PD4	General-purpose I/O	I/O	—	
	PD5	General-purpose I/O	I/O	—	
	PD6	General-purpose I/O	I/O	—	
	PD7	General-purpose I/O	I/O	—	
	PD8	General-purpose I/O	I/O	—	
	PE0	General-purpose I/O	I/O	—	
	PE1	General-purpose I/O	I/O	—	
	PE2	General-purpose I/O	I/O	—	
	PE3	General-purpose I/O	I/O	—	
	PE4	General-purpose I/O	I/O	—	
	PE5	General-purpose I/O	I/O	—	
	PE6	General-purpose I/O	I/O	—	
	PE7	General-purpose I/O	I/O	—	
	PF0	General-purpose I/O	I/O	—	
	PF1	General-purpose I/O	I/O	—	
	PF10	General-purpose I/O	I/O	—	
	PF11	General-purpose I/O	I/O	—	

(continued)

Module	Name	Function	Type	Active	Properties
ARM I/O	PF12	General-purpose I/O	I/O	—	
	PF2	General-purpose I/O	I/O	—	
	PF3	General-purpose I/O	I/O	—	
	PF4	General-purpose I/O	I/O	—	
	PF5	General-purpose I/O	I/O	—	
	PF6	General-purpose I/O	I/O	—	
	PF7	General-purpose I/O	I/O	—	
	USB_VBUS	USB Power Detect Pin	input	—	
ARM Input	RESETn	Reset	input	—	
	USB_VREGI	USB power in	input	—	

5 Product Overview

The XS1-XAU8A-10-FB265 multicore microcontroller combines the flexibility, low latency and determinacy of xCORE, with an integrated ARM Cortex-M3 embedded processor. The devices communicate across the XMOS ARM Bridge. As such, the XS1-XAU8A-10-FB265 can be held in a very low power mode until the ARM core is woken up and used to boot the xCORE tile. Once active the ARM-core can be controlled by any xCORE node on the system.

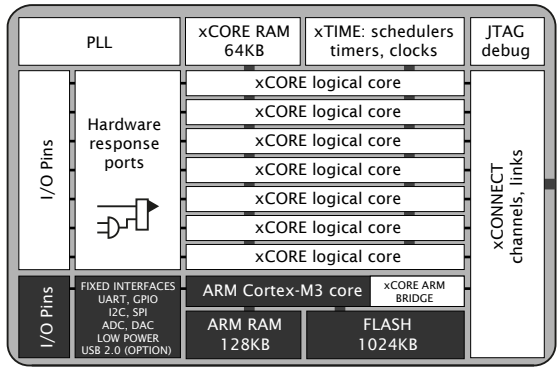


Figure 2: Block Diagram

5.1 xCORE Tile

The xCORE Tile has tightly integrated I/O and on-chip memory. The tile contains multiple logical cores that run simultaneously, each of which is guaranteed a slice of processing power and can execute computational code, control software and I/O interfaces. The logical cores use channels to exchange data within a tile or across tiles. Multiple devices can be deployed and connected using an integrated switching network, enabling more resources to be added to a design. The I/O pins are driven using intelligent ports that can serialize data, interpret strobe signals and wait for scheduled times or events, making the device ideal for real-time control applications.

5.2 xCORE Logical cores

The xCORE tile has up to 8 active logical cores, which issue instructions down a shared four-stage pipeline. One of the cores on the XS1-XAU8A-10-FB265 is dedicated to the xCORE-ARM Bridge (see 6). Instructions from the active cores are issued round-robin. If up to 4 logical cores are active, each core is allocated a quarter of the processing cycles. If more than four logical cores are active, each core is allocated at least $1/n$ cycles (for n cores). Figure 3 shows the guaranteed core performance depending on the number of cores used.

There is no way that the performance of a logical core can be reduced below these predicted levels. Because cores may be delayed on I/O, however, their unused processing cycles can be taken by other cores. This means that for more than

Figure 3: Logical core performance

Speed Grade, MIPS, and frequency	Minimum MIPS per core (for *n* cores)							
	1 (xCORE-ARM)	2	3	4	5	6	7	8
5: 500 MIPS, 500 MHz	125	125	125	125	100	83	71	63

four logical cores, the performance of each core is often higher than the predicted minimum.

5.3 xTIME scheduler

The xTIME scheduler handles the events generated by xCORE Tile resources, such as channels, timers and I/O pins. It ensures that all events are serviced and synchronized with tasks running on separate xCORE Tiles, without the need for an RTOS. Events that occur at the I/O pins are handled by the Hardware-Response ports and fed directly to the appropriate xCORE Tile.

Each task running on its own logical core has the highest priority from the xTIME scheduler, although low priority tasks can share a processing core using cooperative multitasking.

5.4 Hardware Response Ports

Hardware Response ports connect an xCORE tile to one or more physical pins and as such define the interface between hardware attached to an xCORE multicore microcontroller and software running on the xCORE device. The XS1-XAU8A-10-FB265 includes a combination of 1bit, 4bit, 8bit and 16bit ports. In addition, wider ports are partially or fully bonded out making the connected pins available for I/O or xCONNECT links. All pins of a port provide either output or input. Signals in different directions cannot be mapped onto the same port.

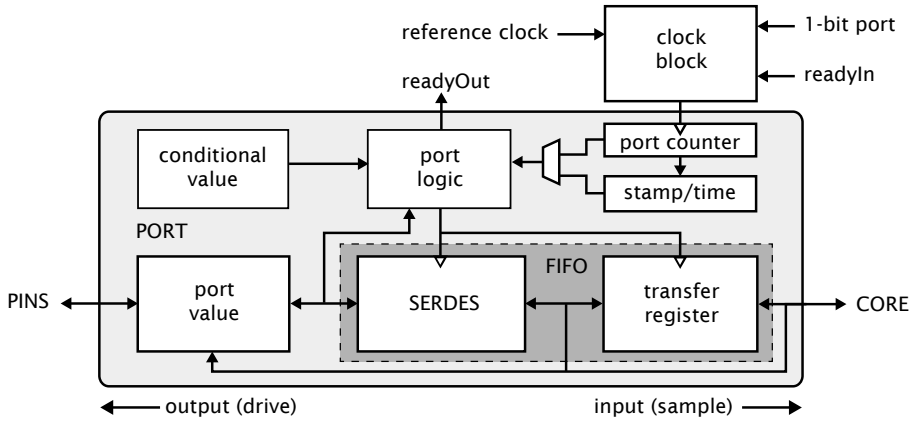


Figure 4: Port block diagram

The ports and links are multiplexed, allowing the pins to be configured for use by ports of different widths or links. If an xConnect Link is enabled, the pins of the

underlying ports are disabled. If a port is enabled, it overrules ports with higher widths that share the same pins. The pins on the wider port that are not shared remain available for use when the narrower port is enabled. Ports always operate at their specified width, even if they share pins with another port.

The port logic can drive its pins high or low, or it can sample the value on its pins, optionally waiting for a particular condition. Ports are accessed using dedicated instructions that are executed in a single processor cycle.

Data is transferred between the pins and core using a FIFO that comprises a SERDES and transfer register, providing options for serialization and buffered data.

Each port has a 16-bit counter that can be used to control the time at which data is transferred between the port value and transfer register. The counter values can be obtained at any time to find out when data was obtained, or used to delay I/O until some time in the future.

The port counter value is automatically saved as a timestamp, that can be used to provide precise control of response times.

5.5 Clock blocks

xCORE devices include a set of programmable clocks called clock blocks that can be used to govern the rate at which ports execute. Each xCORE tile has six clock blocks: the first clock block provides the tile reference clock and runs at a default frequency of 100MHz; the remaining clock blocks can be set to run at different frequencies.

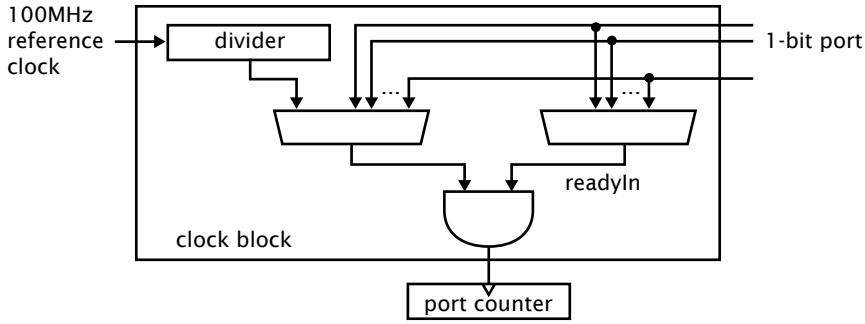


Figure 5: Clock block diagram

A clock block can use a 1-bit port as its clock source allowing external application clocks to be used to drive the input and output interfaces.

In many cases I/O signals are accompanied by strobing signals. The xCORE ports can input and interpret strobe (known as readyIn and readyOut) signals generated by external sources, and ports can generate strobe signals to accompany output data.

On reset, each port is connected to clock block 0, which runs from the xCORE Tile reference clock.

5.6 xCORE Channels and Channel Ends

Logical cores communicate using point-to-point channel connections, formed between two channel ends. A channel-end is a resource on an xCORE tile, that is allocated by the program. Each channel-end has a unique system-wide identifier that comprises a unique number and their tile identifier. Data is transmitted to a channel-end by a sequence of output-instructions; and the other side executes a series of input-instructions. Data can be passed synchronously or asynchronously between the channel ends.

5.7 xCONNECT Switch and Links

XMOS devices provide a scalable architecture, where multiple xCORE devices can be connected together to form one system. Each xCORE device has an xCONNECT interconnect that provides a communication infrastructure for all tasks that run on the various xCORE tiles on the system.

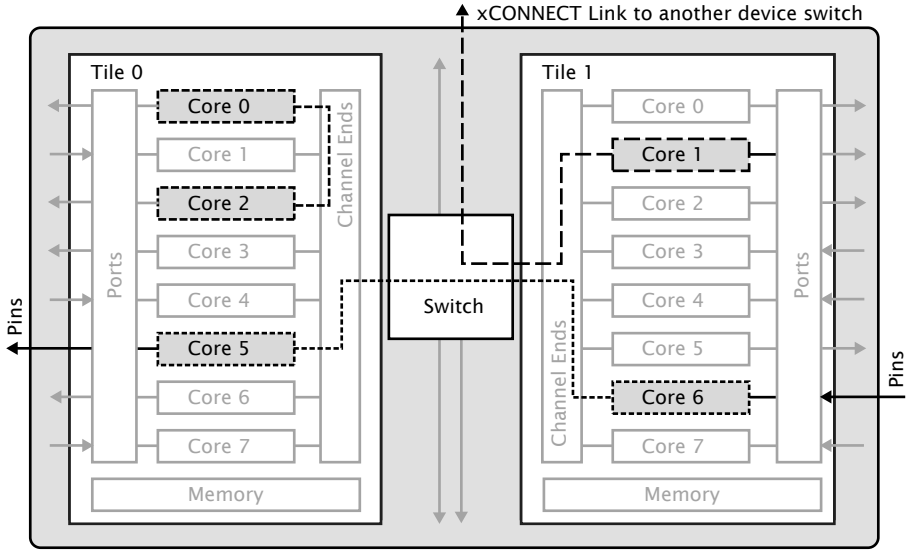


Figure 6: Switch, links and channel ends

The interconnect relies on a collection of switches and xCONNECT links. Each xCORE device has an on-chip switch that can set up circuits or route data. The switches are connected by xCONNECT Links. An xCONNECT link provides a physical connection between two switches. The switch has a routing algorithm that supports many different topologies, including lines, meshes, trees, and hypercubes.

The links operate in either 2 wires per direction or 5 wires per direction mode, depending on the amount of bandwidth required. Circuit switched, streaming and packet switched data can both be supported efficiently. Streams provide the fastest possible data rates between xCORE Tiles (up to 250 MBit/s), but each stream requires a single link to be reserved between switches on two tiles. All packet communications can be multiplexed onto a single link.

5.8 xCORE Timers

xCORE timers are 32-bit counters that are relative to the xCORE Tile reference clock. A timer is defined to tick every 10 ns. This value is derived from the reference clock, which is configured to tick at 100 MHz by default.

5.9 ARM Cortex-M3

The XS1-XAU8A-10-FB265 includes an ARM Cortex-M3 32-bit RISC processor with a Memory Protection Unit and a Wake-up Interrupt Controller to handle interrupts triggered while the CPU is asleep.

Memory management is controlled by the Memory System Controller (MSC) and Direct Memory Access Controller (DMA). The DMA performs memory operations independently of the Cortex-M3, reducing the energy consumption and the workload of the processor, and enabling the system to stay in low energy modes when moving data, for instance from the USART to RAM or from the External Bus Interface to a PWM-generating timer.

5.10 ARM-core modules and peripherals

The ARM-core includes modules and fixed peripherals that are driven by the Cortex-M3. They can be configured using the ARM registers.

Figure 7: ARM-core modules

Module/peripheral	Description
Memory System Controller (MSC)	The program memory unit of the ARM-core.
Direct Memory Access Controller (DMA)	Performs memory operations independently of the CPU, reducing the energy consumption and the workload of the CPU.
Reset Management Unit (RMU)	Handles the reset functionality of the ARM-core.
Energy Management Unit (EMU)	Manages the low energy modes in the ARM-core.
Clock Management Unit (CMU)	Controls the oscillators and clocks on-board the ARM-core.
Watchdog (WDOG)	Generates a reset in case of a system failure, to increase application reliability.
Peripheral Reflex System (PRS)	Allows different peripheral modules to communicate directly with each other in a network, without involving the CPU.
External Bus Interface (EBI)	Provides access to external parallel interface devices such as SRAM, FLASH and ADCs. The interface is memory mapped into the address bus of the Cortex-M3. It is used to communicate with xCORE.

Module/peripheral	Description
Universal Serial Bus Controller (USB)	Full-speed USB 2.0 compliant OTG host/device controller.
Inter-Integrated Circuit Interface (I2C)	Interface between the Cortex-M3 and a serial I2C-bus.
Universal Synchronous/ Asynchronous Receiver/ Transmitter (USART)	Full duplex asynchronous UART communication as well as RS-485, SPI, MicroWire and 3-wire. Also interfaces with ISO7816 SmartCards, I2S and IrDA devices.
Universal Asynchronous Receiver/Transmitter (UART)	Full- and half-duplex asynchronous UART communication.
Low Energy Universal Asynchronous Receiver/ Transmitter (LEUART)	Two-way UART communication up to 9600 baud/s using a 32.768 kHz clock.
Timer/Counter (TIMER)	16-bit general purpose Timer with three compare/capture channels for input capture and compare/Pulse-Width Modulation (PWM) output.
Real Time Counter (RTC)	24-bit counter, clocked by a 32.768 kHz crystal oscillator or a 32.768 kHz RC oscillator.
Backup Real Time Counter (BURTC)	32-bit counter, clocked either by a 32.768 kHz crystal oscillator, a 32.768 kHz RC oscillator or a 1 kHz ULFRCO.
Low Energy Timer (LETIMER)	16-bit timer that is available in energy mode EM2 as well as EM1 and EM0.
Pulse Counter (PCNT)	Counts pulses on a single input or to decode quadrature encoded inputs.
Analog Comparator (ACMP)	Compares the voltage of two analog inputs, with a digital output indicating which input voltage is higher.
Voltage Comparator (VCMP)	Monitors the supply voltage from software. An interrupt can be generated when the supply falls below or rises above a programmable threshold.
Analog to Digital Converter (ADC)	SAR ADC with a resolution of up to 12 bits at up to one million samples per second.
Digital to Analog Converter (DAC)	Fully differential rail-to-rail DAC, with 12-bit resolution.
Operational Amplifier (OPAMP)	General purpose amplifier with rail-to-rail differential input and rail-to-rail single ended output.
Low Energy Sensor Interface (LESENSE)	Highly configurable sensor interface with support for up to 16 individually configurable sensors.
Backup Power Domain	Separate power domain containing a Backup Real Time Counter, BURTC, and a set of retention registers.
Advanced Encryption Standard Accelerator (AES)	Performs AES encryption and decryption with 128-bit or 256-bit keys.
General Purpose Input/Output (GPIO)	General Purpose Input/Output (GPIO) pins, which are divided into ports with up to 16 pins each.

Figure 8: ARM-core peripherals

5.11 ARM-core GPIO

The ARM-core has 70 I/O pins for general input and output, or peripheral configuration. The pins are organized as 16-bit ports indicated by letters A-F; the individual pin on each port in indicated by a number from 15 down to 0.

Pin	Port					
	Port A	Port B	Port C	Port D	Port E	Port F
0	-	PB0	PC0	PD0	PE0	PF0
1	-	PB1	PC1	PD1	PE1	PF1
2	-	PB2	PC2	PD2	PE2	PF2
3	-	PB3	PC4	PD3	PE3	PF3
4	-	PB4	PC5	PD4	PE4	PF4
5	-	PB5	PC5	PD5	PE5	PF5
6	-	PB6	PC6	PD6	PE6	PF6
7	PA7	PB7	PC7	PD7	PE7	PF7
8	PA8	PB8	PC8	PD8	-	-
9	PA9	PB9	PC9	-	-	-
10	PA10	PB10	PC10	PD10	-	PF10
11	PA11	PB11	PC11	-	-	PF11
12	PA12	PB12	PC12	-	-	PF12
13	PA13	PB13	PC13	PD13	-	-
14	PA14	-	PC14	PD14	-	-
15	-	PB15	PC15	PD15	-	-

Some of the pins are used to implement the xCORE-ARM bridge (Figure 10) - they are marked as *NC* in the Pin Configuration table (Section 3). These pins are not available when the xCORE-ARM interface is active.

6 xCORE ARM bridge

The xCORE Tile and ARM-core communicate with each other via the XAB library (Figure 9). On the ARM-core the library is accessed by a C-API which manages the DMA controller. The DMA controller can move data without CPU intervention, effectively reducing the energy consumption for the data transfer. On the xCORE Tile the bridge code runs in a separate task that uses a logical core. The applications can connect to the task to send and receive data.

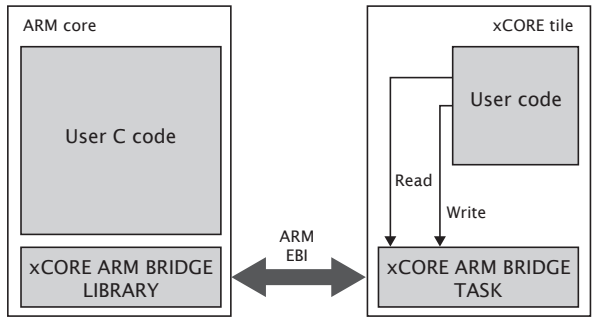


Figure 9: xCORE ARM bridge

See the following application notes for information on using the xCORE-ARM bridge:

- AN00142: *xCORE-XA - xCORE ARM Bridge Library*,
- AN00143: *xCORE-XA - xCORE ARM Bridge Library with DMA*
- AN00144: *xCORE-XA - xCORE ARM Boot Library*

The XAB bridge uses pre-defined ports and one logical core on the xCORE Tile, and predefined signals on the ARM-core, as shown in Figure 10.

7 PLL

The PLL creates a high-speed clock that is used for the switch, tile, and reference clock. The PLL multiplication value is selected through the two MODE pins, and can be changed by software to speed up the tile or use less power. The MODE pins are set as shown in Figure 11:

Figure 11 also lists the values of OD, F and R, which are the registers that define the ratio of the tile frequency to the oscillator frequency:

$$F_{core} = F_{osc} \times \frac{F+1}{2} \times \frac{1}{R+1} \times \frac{1}{OD+1}$$

OD, F and R must be chosen so that $0 \leq R \leq 63$, $0 \leq F \leq 4095$, $0 \leq OD \leq 7$, and $260MHz \leq F_{osc} \times \frac{F+1}{2} \times \frac{1}{R+1} \leq 1.3GHz$. The OD, F, and R values can be modified by writing to the digital node PLL configuration register.

The MODE pins must be held at a static value during and after deassertion of the system reset.

xCORE-XA Pad	xCORE I/O Pin	Port		
		1-bit	8-bit	32-bit
A15	X0D26		P8C0	
B15	X0D27		P8C1	
A14	X0D28		P8C2	
B14	X0D29		P8C3	
A13	X0D30		P8C4	
B13	X0D31		P8C5	
A12	X0D32		P8C6	
B12	X0D33		P8C7	
D18	X0D38	P1I		
D18	X0D39	P1J		
P3	X0D56			P32C7
R2	X0D57			P32C8
R3	X0D58			P32C9
T2	X0D61			P32C10
U1	X0D62			P32C11
U2	X0D63			P32C12
V4	X0D64			P32C13
W4	X0D65			P32C14

Figure 10: Reserved xCORE ARM bridge ports and signals

Oscillator Frequency	MODE 1	MODE 0	Tile Frequency	PLL Ratio	PLL settings OD	F	R
5-13 MHz	0	0	130-399.75 MHz	30.75	1	122	0
13-20 MHz	1	1	260-400.00 MHz	20	2	119	0
20-48 MHz	1	0	167-400.00 MHz	8.33	2	49	0
48-100 MHz	0	1	196-400.00 MHz	4	2	23	0

Figure 11: PLL multiplier values and MODE pins

For 500 MHz parts, once booted, the PLL must be reprogrammed to provide this tile frequency. The XMOS tools perform this operation by default.

Further details on configuring the clock can be found in the XS1-L Clock Frequency Control document, X1433.

8 Boot Procedure

8.1 xCORE-XA boot procedure

The xCORE Tile and ARM-core must be booted separately before they can communicate. Once the xCORE Tile and ARM-core are awake they can communicate using the xCORE ARM bridge- see Section 6.

The ARM-core should be booted first and then the xCORE Tile can be booted from the flash memory contained within the ARM-core using the xCORE-ARM boot library (Figure 12). This allows the XS1-XAU8A-10-FB265 to boot from a single internal flash device.

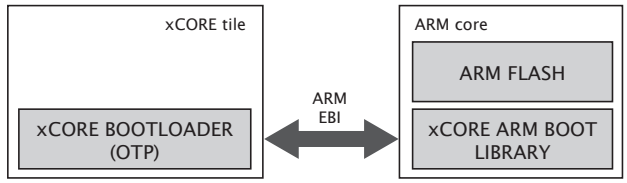


Figure 12: xCORE-XA booting

For information on using the xCORE-ARM boot library see: AN00144: *xCORE-XA - xCORE ARM Boot Library*.

8.2 ARM core boot procedure

The ARM-core boot loader attempts to boot an application in flash or wait for a new firmware update depending on the state of the DBG_SWCLK pin when coming out of RESET. When uploading new firmware, the boot loader decrypts, verifies and stores the new application in flash. It can optionally be configured to make use of a temporary storage area to make sure that the ARM-core contains a valid program.

Figure 13 shows a state diagram of the boot loader. Out of reset the bootloader checks the state of the DBG_SWCLK (this can be configured to use any pin). If the pin is pulled HIGH the ARM-core enters *boot loader mode* and waits for commands over UART. If it receives the upload command it will enter *upload mode* where it can accept new firmware.

In *upload mode* the bootloader receives data packets, decrypts them if necessary and writes them to internal flash. This allows developers to upload new encrypted firmware.

If the DBG_SWCLK pin is low when the ARM-core comes out of reset, the bootloader will attempt to boot the application in flash. The boot loader will first verify the application by checking the verified flag in the firmware header. If temporary storage is enabled and the application in the boot area fails verification the boot loader will check the temporary storage. If it finds a valid application here it will start copying it to the boot area and then boot it. When there are no valid applications in memory the boot loader will enter a *low power wait mode* and wait for the boot loader pin to be pulled HIGH.

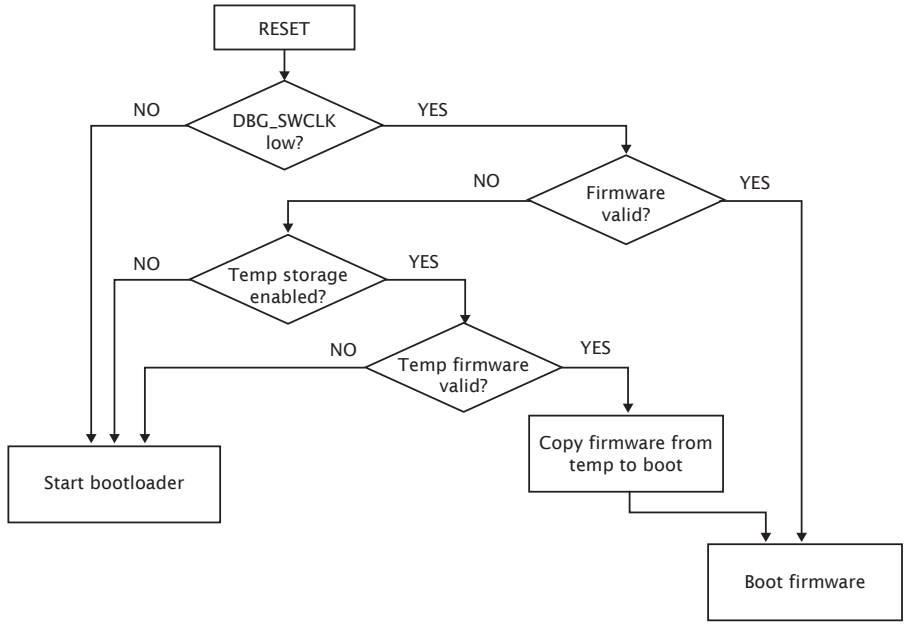


Figure 13: ARM bootloader

8.3 ARM-core energy modes

The ARM-core can run in five different energy modes. Instead of running the core in active mode all the time, it can be switched to a mode that is more energy efficient. The peripherals are available in various energy modes. For information on how to use the different modes see *Energy Management Unit* - Section K.

Mode	Name	Description	Power Consumption
0	Active/Run	High performance CPU and peripherals designed for ultra-low power operation	180 µA/MHz
1	Sleep	Stay in low energy modes while performing advanced tasks	45 µA/MHz
2	Deep Sleep	Advanced low power and autonomous operation without CPU intervention	0.9 µA
3	Stop	Operation, full RAM retention and short 2µs wake-up from interrupts	0.6 µA
4	Shutoff	For applications that do not need RTC or RAM retention	20 nA

Figure 14: ARM-core energy modes

8.4 xCORE Tile boot procedure

The xCORE Tile is kept in reset by driving RST_N low. When in reset, all GPIO pins are high impedance. When the device is taken out of reset by releasing RST_N the processor starts its internal reset process. After 15-150 μs (depending on the input clock), all GPIO pins have their internal pull-resistor enabled, and the processor boots at a clock speed that depends on MODE0 and MODE1.

The xCORE Tile boot procedure is illustrated in Figure 15. In normal usage, MODE[3:2] controls the boot source according to the table in Figure 16.

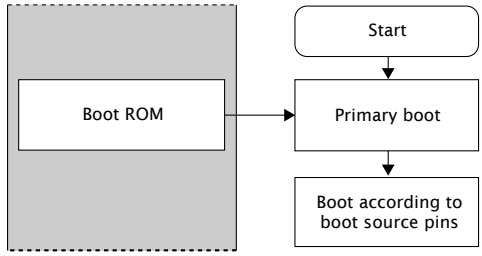


Figure 15: Boot procedure

MODE[3]	MODE[2]	Boot Source
0	0	None: Device waits to be booted via JTAG
0	1	Reserved
1	0	Boot from ARM-core
1	1	SPI

Figure 16: Boot source pins

The boot image has the following format:

- A 32-bit program size s in words.
- Program consisting of $s \times 4$ bytes.
- A 32-bit CRC, or the value 0x0D15AB1E to indicate that no CRC check should be performed.

The program size and CRC are stored least significant byte first. The program is loaded into the lowest memory address of RAM, and the program is started from that address. The CRC is calculated over the byte stream represented by the program size and the program itself. The polynomial used is 0xEDB88320 (IEEE 802.3); the CRC register is initialized with 0xFFFFFFFF and the residue is inverted to produce the CRC.

8.5 xCORE Tile boot from SPI

If set to boot from SPI, the xCORE processor enables the four pins specified in Figure 17, and drives the SPI clock at 2.5 MHz (assuming a 400 MHz core clock). A READ command is issued with a 24-bit address 0x000000. The clock polarity and phase are 0 / 0.

Pin	Signal	Description
X0D00	MISO	Master In Slave Out (Data)
X0D01	SS	Slave Select
X0D10	SCLK	Clock
X0D11	MOSI	Master Out Slave In (Data)

Figure 17: SPI pins

The xCORE Tile expects each byte to be transferred with the *least-significant bit first*. Programmers who write bytes into an SPI interface using the most significant bit first may have to reverse the bits in each byte of the image stored in the SPI device.

If a large boot image is to be read in, it is faster to first load a small boot-loader that reads the large image using a faster SPI clock, for example 50 MHz or as fast as the flash device supports.

The pins used for SPI boot are hardcoded in the boot ROM and cannot be changed. If required, an SPI boot program can be burned into OTP that uses different pins.

9 Memory

The xCORE Tile and ARM-core each has an SRAM module for storing application code and data. The ARM-core has integrated flash memory for storing program code, user data and flash lock bits - see Figure 18.

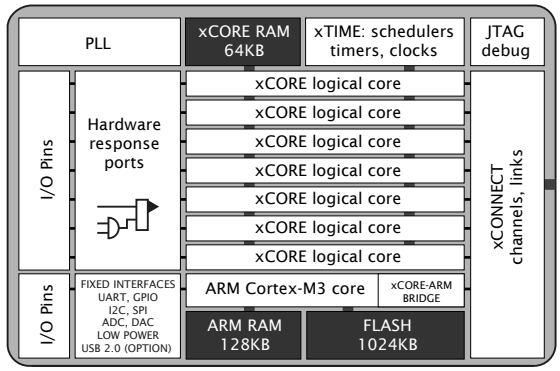


Figure 18: xCORE-XA memory

9.1 OTP

The xCORE Tile integrates 8 KB one-time programmable (OTP) memory along with a security register that configures system wide security features. The OTP holds data in four sectors each containing 512 rows of 32 bits which can be used to implement secure bootloaders and store encryption keys. Data for the security register is loaded from the OTP on power up. All additional data in OTP is copied from the OTP to SRAM and executed first on the processor.

The OTP memory is programmed using three special I/O ports: the OTP address port is a 16-bit port with resource ID 0x100200, the OTP data is written via a 32-bit port with resource ID 0x200100, and the OTP control is on a 16-bit port with ID 0x100300. Programming is performed through `libotp` and `xburn`.

9.2 SRAM

The xCORE Tile integrates a single 64 KB SRAM bank for both instructions and data. All internal memory is 32 bits wide, and instructions are either 16-bit or 32-bit. Byte (8-bit), half-word (16-bit) or word (32-bit) accesses are supported and are executed within one tile clock cycle. There is no dedicated external memory interface for the xCORE Tile although data memory can be expanded through appropriate use of the ports.

The ARM-core has a 128 KB SRAM bank for application data. Instructions can be executed from SRAM, and the DMA may be used to transfer data between the SRAM, Flash memory and peripherals. The SRAM is divided into 32 KB blocks that can be individually powered down when not in use. It supports bit-band access support, and data retention of the entire memory in modes EM0 to EM3.

9.3 Flash memory

The ARM-core flash memory is readable and writable from both the Cortex-M3 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.

Read and write operations are supported in the energy modes EM0 and EM1.

10 JTAG

10.1 xCORE Tile

The xCORE JTAG module can be used for loading programs, boundary scan testing, in-circuit source-level debugging and programming the OTP memory.

The JTAG chain structure is illustrated in Figure 19. Directly after reset, two Test Access Points (TAP) controllers are present in the JTAG chain: the boundary scan TAP and the chip TAP. The boundary scan TAP is a standard 1149.1 compliant TAP that can be used for boundary scan of the I/O pins. The chip TAP provides access into the xCORE Tile, switch and OTP for loading code and debugging.

The JTAG module can be reset by holding TMS high for five clock cycles.

The DEBUG_N pin is used to synchronize the debugging of multiple xCORE Tiles. This pin can operate in both output and input mode. In output mode and when configured to do so, DEBUG_N is driven low by the device when the processor hits

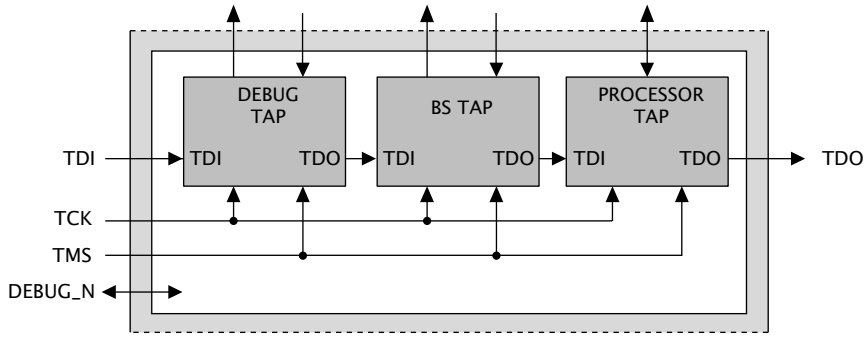


Figure 19: JTAG chain structure

a debug break point. Prior to this point the pin will be tri-stated. In input mode and when configured to do so, driving this pin low will put the xCORE Tile into debug mode. Software can set the behavior of the xCORE Tile based on this pin. This pin should have an external pull up of 4K7-47KΩ or left not connected in single core applications.

The JTAG device identification register can be read by using the IDCODE instruction. Its contents are specified in Figure 20.

Figure 20: IDCODE return value

Bit31																															Bit0
Version				Part Number																Manufacturer Identity											1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1	0	0	1	1
0				0				0				0				?				6				3				3			

The JTAG usercode register can be read by using the USERCODE instruction. Its contents are specified in Figure 21. The OTP User ID field is read from bits [22:31] of the security register , *see* §9.1 (all zero on unprogrammed devices).

Figure 21: USERCODE return value

Bit31																															Bit0
OTP User ID										Unused				Silicon Revision																	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0				0				0				2				8				0				0				0			

10.2 ARM-core debug interface

The xCORE-XA ARM core includes hardware debug support through a 2-pin serial-wire debug interface and an Embedded Trace Module (ETM) for data/instruction tracing. In addition there is also a 1-wire Serial Wire Viewer pin which can be used to output profiling information, data trace and software-generated messages.

11 Board Integration

The device has the following power supply pins:

- VDDIO pins for the I/O lines
- VDD pins for the core voltage supply
- PLLVDD pins for the PLL
- AVDD pins for the ARM core
- VDDEM pins for the ARM core

Several pins of each type are provided to minimize the effect of inductance within the package, all of which must be connected. The power supplies must be brought up monotonically and input voltages must not exceed specification at any time.

The supply must ramp to its final value before VDD reaches 0.4 V.

The following ground pins are provided:

- GND for all supplies

All ground pins must be connected directly to the board ground.

RST_N is an active-low asynchronous-assertion global reset signal. Following a reset, the PLL re-establishes lock after which the device boots up according to the boot mode (*see* §8). RST_N must be asserted low during and after power up for 100 ns.

11.1 Land patterns and solder stencils

The land pattern recommendations in this document are based on a RoHS compliant process and derived, where possible, from the nominal *Generic Requirements for Surface Mount Design and Land Pattern Standards* IPC-7351B specifications. This standard aims to achieve desired targets of heel, toe and side fillets for solder-joints.

Solder paste and ground via recommendations are based on our engineering and development kit board production. They have been found to work and optimized as appropriate to achieve a high yield. The size, type and number of vias used in the center pad affects how much solder wicks down the vias during reflow. This in turn, along with solder paste coverage, affects the final assembled package height. These factors should be taken into account during design and manufacturing of the PCB.

The following land patterns and solder paste contains recommendations. Final land pattern and solder paste decisions are the responsibility of the customer. These should be tuned during manufacture to suit the manufacturing process.

The package is a 265 pin Fine Ball Grid Array package on a 0.8mm pitch with 0.4mm balls.

An example land pattern is shown in Figure 22.

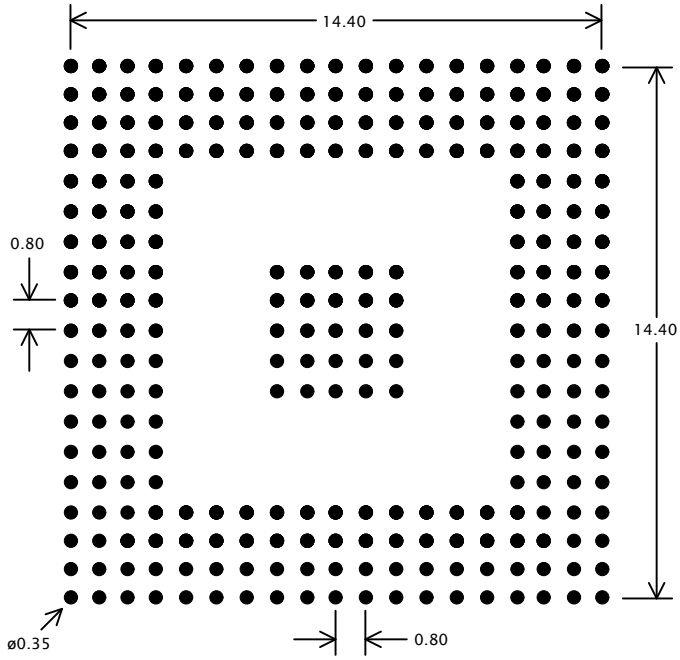


Figure 22: Example land pattern

Pad widths and spacings are such that solder mask can still be applied between the pads using standard design rules. This is highly recommended to reduce solder shorts.

11.2 Ground and Thermal Vias

Vias next to each ground ball into the ground plane of the PCB are recommended for a low inductance ground connection and good thermal performance. Vias with a 0.6mm diameter annular ring and a 0.3mm drill would be suitable.

11.3 Moisture Sensitivity

XMOS devices are, like all semiconductor devices, susceptible to moisture absorption. When removed from the sealed packaging, the devices slowly absorb moisture from the surrounding environment. If the level of moisture present in the device is too high during reflow, damage can occur due to the increased internal vapour pressure of moisture. Example damage can include bond wire damage, die lifting, internal or external package cracks and/or delamination.

All XMOS devices are Moisture Sensitivity Level (MSL) 3 - devices have a shelf life of 168 hours between removal from the packaging and reflow, provided they are stored below 30C and 60% RH. If devices have exceeded these values or an

included moisture indicator card shows excessive levels of moisture, then the parts should be baked as appropriate before use. This is based on information from *Joint IPC/JEDEC Standard For Moisture/Reflow Sensitivity Classification For Nonhermetic Solid State Surface-Mount Devices* J-STD-020 Revision D.

12 xCORE DC and Switching Characteristics

12.1 Operating Conditions

Figure 23: Operating conditions

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
VDD	Tile DC supply voltage	0.95	1.00	1.05	V	
VDDIO	I/O supply voltage	3.00	3.30	3.60	V	
PLL_AVDD	PLL analog supply	0.95	1.00	1.05	V	
OTP_VCC	OTP supply voltage	3.00	3.30	3.60	V	
OTP_VPP	OTP external programming voltage (optional program only)	6.18	6.50	6.83	V	
Cl	xCORE Tile I/O load capacitance			25	pF	
Ta	Ambient operating temperature (Commercial)	0		70	°C	
	Ambient operating temperature (Industrial)	-40		85	°C	
Tj	Junction temperature			125	°C	
Tstg	Storage temperature	-65		150	°C	

12.2 DC Characteristics

Figure 24: DC characteristics

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
V(IH)	Input high voltage	2.00		3.60	V	A
V(IL)	Input low voltage	-0.30		0.70	V	A
V(OH)	Output high voltage	2.00			V	B, C
V(OL)	Output low voltage			0.60	V	B, C
R(PU)	Pull-up resistance		35K		Ω	D
R(PD)	Pull-down resistance		35K		Ω	D

A All pins except power supply pins.

B Ports 1A, 1D, 1E, 1H, 1I, 1J, 1K and 1L are nominal 8 mA drivers, the remainder of the general-purpose I/Os are 4 mA.

C Measured with 4 mA drivers sourcing 4 mA, 8 mA drivers sourcing 8 mA.

D Used to guarantee logic state for an I/O when high impedance. The internal pull-ups/pull-downs should not be used to pull external circuitry.

12.3 ESD Stress Voltage

Figure 25: ESD stress voltage

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
HBM	Human body model	-2.00		2.00	KV	
MM	Machine model	-200		200	V	

12.4 Reset Timing

Figure 26: Reset timing

Symbol	Parameters	MIN	TYP	MAX	UNITS	Notes
T(RST)	Reset pulse width	5			us	
T(INIT)	Initialization time			150	µs	A

A Shows the time taken to start booting after RST_N has gone high.

12.5 Power Consumption

Figure 27: xCORE Tile currents

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
I(DDCQ)	Quiescent VDD current		14		mA	A, B, C
PD	Tile power dissipation		450		µW/MIPS	A, D, E, F
IDD	Active VDD current		200	375	mA	A, G
I(ADDPLL)	PLL_AVDD current			7	mA	H

A Use for budgetary purposes only.

B Assumes typical tile and I/O voltages with no switching activity.

C Includes PLL current.

D Assumes typical tile and I/O voltages with nominal switching activity.

E Assumes 1 MHz = 1 MIPS.

F PD(TYP) value is the usage power consumption under typical operating conditions.

G Measurement conditions: VDD = 1.0 V, VDDIO = 3.3 V, 25 °C, 500 MHz, average device resource usage.

H PLL_AVDD = 1.0 V

The tile power consumption of the device is highly application dependent and should be used for budgetary purposes only.

More detailed power analysis can be found in the XS1-XAU Power Consumption document,

12.6 Clock

Figure 28: Clock

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
f	Frequency	4.22	20	100	MHz	
SR	Slew rate	0.10			V/ns	
TJ(LT)	Long term jitter (pk-pk)			2	%	A
f(MAX)	Processor clock frequency			500	MHz	B

A Percentage of CLK period.

B Assumes typical tile and I/O voltages with nominal activity.

Further details can be found in the XS1-XAU Clock Frequency Control document,

12.7 xCORE Tile I/O AC Characteristics

Figure 29: I/O AC characteristics

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
T(XOVALID)	Input data valid window	8			ns	
T(XOINVALID)	Output data invalid window	9			ns	
T(XIFMAX)	Rate at which data can be sampled with respect to an external clock			60	MHz	

The input valid window parameter relates to the capability of the device to capture data input to the chip with respect to an external clock source. It is calculated as the sum of the input setup time and input hold time with respect to the external clock as measured at the pins. The output invalid window specifies the time for which an output is invalid with respect to the external clock. Note that these parameters are specified as a window rather than absolute numbers since the device provides functionality to delay the incoming clock with respect to the incoming data.

Information on interfacing to high-speed synchronous interfaces can be found in the XS1 Port I/O Timing document, X5821.

12.8 xConnect Link Performance

Figure 30: Link performance

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
B(2blinkP)	2b link bandwidth (packetized)			87	MBit/s	A, B
B(5blinkP)	5b link bandwidth (packetized)			217	MBit/s	A, B
B(2blinkS)	2b link bandwidth (streaming)			100	MBit/s	B
B(5blinkS)	5b link bandwidth (streaming)			250	MBit/s	B

A Assumes 32-byte packet in 3-byte header mode. Actual performance depends on size of the header and payload.

B 7.5 ns symbol time.

The asynchronous nature of links means that the relative phasing of CLK clocks is not important in a multi-clock system, providing each meets the required stability criteria.

12.9 JTAG Timing

Figure 31: JTAG timing

Symbol	Parameter	MIN	TYP	MAX	UNITS	Notes
f(TCK_D)	TCK frequency (debug)			18	MHz	
f(TCK_B)	TCK frequency (boundary scan)			10	MHz	
T(SETUP)	TDO to TCK setup time	5			ns	A
T(HOLD)	TDO to TCK hold time	5			ns	A
T(DELAY)	TCK to output delay			15	ns	B

A Timing applies to TMS and TDI inputs.

B Timing applies to TDO output from negative edge of TCK.

13 ARM core DC and Switching Characteristics

13.1 Power consumption

Symbol	Parameter	Condition	Min	Typ	Max	Unit
IEM0	EM0 current. No prescaling. Running prime number calculation code from Flash.	32 MHz HFXO, all peripheral clocks disabled, VDD=3v0		200		µA/MHz
		28 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		201	261	µA/MHz
		21 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		203	263	µA/MHz
		14 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		204	270	µA/MHz
		11 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		207	273	µA/MHz
		6.6 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		212	282	µA/MHz
		1.2 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		244		µA/MHz
IEM1	EM1 current	32 MHz HFXO, all peripheral clocks disabled, VDD=3v0		50		µA/MHz
		28 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		52	69	µA/MHz
		21 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		53	71	µA/MHz
		14 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		56	77	µA/MHz
		11 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		57	80	µA/MHz
		6.6 MHz HFRCO, all peripheral clocks disabled, VDD=3v0		62	92	µA/MHz
		1.2 MHz HFRCO. all peripheral clocks disabled, VDD=3v0		114		µA/MHz
IEM2	EM2 current	EM2 current with RTC at 1Hz, RTC prescaled to 1kHz, 32.768 kHz LFRCO, VDD=3v0, TAMB=25°C		1.1		µA
		EM2 current with RTC at 1Hz, RTC prescaled to 1kHz, 32.768 kHz LFRCO, VDD=3v0, TAMB=85°C		4.0	8.0	µA
IEM3	EM3 current	VDD=3v0, TAMB=25°C		0.9		µA
		VDD=3v0, TAMB=85°C		3.8	7.8	µA
IEM4	EM4 current	VDD=3v0, TAMB=25°C		0.02		µA
		VDD=3v0, TAMB=85°C		0.25	0.7	µA

13.2 Transition between energy modes

Symbol	Parameter	Min	Typ	Max	Unit
tEM10	Transition time from EM1 to EM0		01		HF core CLK cycles
tEM20	Transition time from EM2 to EM0		2		µs
tEM30	Transition time from EM3 to EM0		2		µs
tEM40	Transition time from EM4 to EM0		163		µs

13.3 Current consumption

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VBODextthr-	BOD threshold on falling external supply voltage		1.82		1.85	V
VBODintthr-	BOD threshold on falling internally regulated supply voltage		1.62		1.68	V
VBODextthr+	BOD threshold on rising external supply voltage				1.85	V
VPORthr+	Power-on Reset (POR) threshold on rising external supply voltage				1.98	V
tRESET	Delay from reset is re- leased until program execution starts	Applies to Power-on Reset, Brown-out Reset and pin reset.		163		us
CDECOUPLE	Voltage regulator decoupling capacitor.	X5R capacitor recommended. Apply between DECOUPLE pin and GROUND		1		uF
CUSB_VREGO	USB voltage regulator out decoupling capacitor.	X5R capacitor recommended. Apply between USB_VREGO pin and GROUND		1		uF
CUSB_VREGI	USB voltage regulator in decoupling capacitor.	X5R capacitor recommended. Apply between USB_VREGI pin and GROUND		4.7		uF

13.4 Flash

Symbol	Parameter	Condition	Min	Typ	Max	Unit
ECFLASH	Flash erase cycles before failure		20000			cycles
RETFLASH	Flash data retention	TAMB<150°C TAMB<85°C TAMB<70°C	10000 10 20			h years years
tW_PROG	Word (32-bit) programming time		20			µs
tPERASE	Page erase time	< 512KB >= 512KB, LPERASE == 0 >= 512KB, LPERASE == 1	20 20 40	20.4 20.4 40.4	20.8 20.8 40.8	ms ms ms
tDERASE	Device erase time	< 512KB >= 512KB	40	40.8	41.6 161.6	ms ms
IERASE	Erase current	< 512KB >= 512KB, LPERASE == 0 >= 512KB, LPERASE == 1			71 141 71	mA mA mA
IWRITE	Write current	< 512KB >= 512KB, LPWRITE == 0 >= 512KB, LPWRITE == 1			71 141 71	mA mA mA
VFLASH	Supply voltage during flash erase and write		1.8		3.8	V

13.5 GPIO

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VIOIL	Input low voltage				0.3VDD	V
VIOIH	Input high voltage		0.7VDD			V
VIOOH	Output high voltage	Sourcing 6 mA, VDD=1.8V, GPIO_Px_CTRL DRIVE-MODE = STANDARD Sourcing 6 mA, VDD=3.0V, GPIO_Px_CTRL DRIVE-MODE = STANDARD Sourcing 20 mA, VDD=1.8V, GPIO_Px_CTRL DRIVE-MODE = HIGH Sourcing 20 mA, VDD=3.0V, GPIO_Px_CTRL DRIVE-MODE = HIGH	0.75VDD 0.95VDD 0.7VDD 0.9VDD			V V V V
VIOOL	Output low voltage	Sinking 6 mA, VDD=1.8V, GPIO_Px_CTRL DRIVE-MODE = STANDARD Sinking 6 mA, VDD=3.0V, GPIO_Px_CTRL DRIVE-MODE = STANDARD Sinking 20 mA, VDD=1.8V, GPIO_Px_CTRL DRIVE-MODE = HIGH Sinking 20 mA, VDD=3.0V, GPIO_Px_CTRL DRIVE-MODE = HIGH			0.25VDD 0.05VDD 0.3VDD 0.1VDD	V V V V
IIOLEAK	Input leakage current	High Impedance IO connected to GROUND or Vdd			+/-25	nA
RPU	I/O pin pull-up resistor			40		kOhm
RPD	I/O pin pull-down resistor			40		kOhm
RIOESD	Internal ESD series resistor			200		Ohm
tIOGLITCH	Pulse width of pulses to be removed by the glitch suppression filter		10		50	ns
tIOOF	Output fall time	0.5 mA drive strength and load capacitance CL=12.5-25pF. 2mA drive strength and load capacitance CL=350-600pF	20+0.1CL 20+0.1CL		250 250	ns ns
VIOHYST	I/O pin hysteresis (VIOTHR+ - VIOTHR-)	VDD = 1.8 - 3.8 V	0.1VDD			V

13.6 Oscillators

13.6.1 LFXO

Symbol	Parameter	Condition	Min	Typ	Max	Unit
fLFXO	Supported nominal crystal frequency			32.768		kHz
ESRLFXO	Supported crystal equivalent series resistance (ESR)			30	120	kOhm
CLFXOL	Supported crystal external load range		5		25	pF
DCLFXO	Duty cycle		48	50	53.5	%
ILFXO	Current consumption for core and buffer after startup.	ESR=30 kOhm, CL=10 pF, LFXOBOOST in CMU_CTRL is 1		190		nA
tLFXO	Start-up time.	ESR=30 kOhm, CL=10 pF, 40% - 60% duty cycle has been reached, LFXOBOOST in CMU_CTRL is 1		400		ms

For safe startup of a given crystal, the load capacitance should be larger than the value indicated below for a given LFXOBOOST setting. The minimum supported load capacitance depends on the crystal shunt capacitance, C0, which is specified in crystal vendors' datasheet.

Symbol	Capacitance [pF]															
Shunt Capacitance C0	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
CLmin lfxoboost = 0 redlfxoboost = 1	3.7	4.0	4.3	4.5	4.8	5.0	5.3	5.5	5.7	5.9	6.0	6.2	6.4	6.5	6.7	6.9
CLmin lfxoboost = 1 redlfxoboost = 0	7.3	7.7	8.2	8.6	9.0	9.3	9.6	10.0	10.3	10.5	10.8	11.1	11.3	11.6	11.8	12.1
CLmin lfxoboost = 1 redlfxoboost = 1	10.0	10.6	11.1	11.6	12.1	12.6	13.0	13.4	13.8	14.1	14.5	14.8	15.1	15.4	15.7	16.0
CLmin lfxoboost = 1 redlfxoboost = 0	12.5	13.2	13.9	14.5	15.0	15.5	16.0	16.5	16.9	17.4	17.8	18.2	18.5	18.9	19.3	19.6

13.6.2 HFXO

Symbol	Parameter	Condition	Min	Typ	Max	Unit
fHFXO	Supported nominal crystal Frequency		4		48	MHz
ESRHFXO	Supported crystal equivalent series resistance (ESR)	Crystal frequency 32 MHz Crystal frequency 4 MHz		30 400	60 1500	Ohm Ohm
gmHFXO	The transconductance of the HFXO input transistor at crystal startup	HFXOBOOST in CMU_CTRL equals 0b11	20			mS
CHFXOL	Supported crystal external load range		5		25	pF
DCHFXO	Duty cycle		46	50	54	%
IHFXO	Current consumption for HFXO after startup	4 MHz: ESR=400 Ohm, CL=20 pF, HFXOBOOST in CMU_CTRL equals 0b11 32 MHz: ESR=30 Ohm, CL=10 pF, HFXOBOOST in CMU_CTRL equals 0b11		85 165		µA µA
tHFXO	Startup time	32 MHz: ESR=30 Ohm, CL=10 pF, HFXOBOOST in CMU_CTRL equals 0b11		400		µs

13.6.3 LFRCO

Symbol	Parameter	Condition	Min	Typ	Max	Unit
fLFRCO	Oscillation frequency , VDD= 3v0, TAMB=25°C			32.768		kHz
tLFRCO	Startup time not including software calibration			150		µs
ILFRCO	Current consumption			190		nA
TUNESTEPL-FRCO	Frequency step for LSB change in TUNING value			1.5		%

13.6.4 HFRCO

Symbol	Parameter	Condition	Min	Typ	Max	Unit
fHFRCO	Oscillation frequency, VDD=3v0, TAMB=25°C	28 MHz frequency band		28		MHz
		21 MHz frequency band		21		MHz
		14 MHz frequency band		14		MHz
		11 MHz frequency band		11		MHz
		7 MHz frequency band		6.6		MHz
		1 MHz frequency band		1.2		MHz
tHFRCO_settling	Settling time after start-up	fHFRCO = 14 MHz		0.6		Cycles
IHFRCO	Current consumption	fHFRCO = 28 MHz		106		µA
		fHFRCO = 21 MHz		93		µA
		fHFRCO = 14 MHz		77		µA
		fHFRCO = 11 MHz		72		µA
		fHFRCO = 6.6 MHz		63		µA
		fHFRCO = 1.2 MHz		22		µA
DCHFRCO	Duty cycle	fHFRCO = 14 MHz	48.5	50	51	%
TUNESTEPH- FRCO	Frequency step for LSB change in TUNING value			0.3		%

13.6.5 ULFRCO

Symbol	Parameter	Condition	Min	Typ	Max	Unit
fULFRCO	Oscillation frequency	25°C, 3V	0.8		1.5	kHz
TCULFRCO	Temperature coefficient				0.05	%/°C
VCULFRCO	Supply voltage coefficient			-18.2		%/V

13.7 ADC

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VADCIN	Input voltage range	Single ended Differential	0 -VREF/2		VREF VREF/2	V V
VADCREFIN	Input range of external reference voltage, single ended and differential		1.25		VDD	V
VADCREFIN_CH7	Input range of external negative reference voltage on channel 7	See VADCREFIN	0		VDD-1.1	V
VADCREFIN_CH6	Input range of external positive reference voltage on channel 6	See VADCREFIN	0.625		VDD	V
VADCCMIN	Common mode input range		0		VDD	V
IADCIN	Input current	2pF sampling capacitors			<100	nA
CMRRADC	Analog input common mode rejection ratio			65		dB
IADC	Average active current	1 MSamples/s, 12 bit, external reference 10 kSamples/s 12 bit, internal 1v25 reference, WARMUP-MODE in ADCn_CTRL set to 0b00 10 kSamples/s 12 bit, internal 1v25 reference, WARMUP-MODE in ADCn_CTRL set to 0b01 10 kSamples/s 12 bit, internal 1v25 reference, WARMUP-MODE in ADCn_CTRL set to 0b10		351 63 64	 67 	µA µA µA µA
IADCREF	Current consumption of internal voltage reference	Internal voltage reference		65		µA
CADCIN	Input capacitance			2		pF
RADCIN	Input ON resistance		1			MOhm
RADCFILT	Input RC filter resistance			10		kOhm
CADCFILT	Input RC filter/decoupling capacitance			250		fF
fADCCLK	ADC Clock Freq				13	MHz
tADCCONV	Conversion time	6 bit 10 bit 12 bit	7	 11 13		ADC-CLK Cycles ADC-CLK Cycles ADC-CLK Cycles
tADCACQ	Acquisition time	Programmable	1		256	ADC-CLK Cycles
tADCACQVDD3	Required acquisition time for VDD/3 reference		2			µs
tADCSTART	Startup time of reference generator and ADC core in NORMAL mode Startup time of reference generator and ADC core in KEEPADCWARM mode			 1	5	µs µs
VADCOFFSET	Offset voltage	After calibration, single ended After calibration, differential			0.3 0.3	mV mV
TGRADADCTH	Thermometer output gradient			-1.92 -6.3		mV/°C ADC Codes/°C
DNLADC	Differential non-linearity (DNL)			± 0.7		LSB
INLADC	Integral non-linearity (INL), End point method				± 1.2	LSB
MCADC	No missing codes		11.999[1]		12	bits
GAINED	Gain error drift	1v25 reference 2v5 reference		0.01[2] 0.01[2]	0.033[3] 0.03[3]	%/°C %/°C
OFFSETED	Offset error drift	1v25 reference 2v5 reference		 0.2[2]	0.2[2] 0.62[3]	0.7[3] LSB/°C LSB/°C

continued ...

Symbol	Parameter	Condition	Min	Typ	Max	Unit
SNRADC	Signal to Noise Ratio (SNR)	1 MSamples/s, 12 bit, single ended, internal 1v25 reference		59		dB
		1 MSamples/s, 12 bit, single ended, internal 2v5 reference		63		dB
		1 MSamples/s, 12 bit, single ended, VDD reference		65		dB
		1 MSamples/s, 12 bit, differential, internal 1v25 reference		60		dB
		1 MSamples/s, 12 bit, differential, internal 2v5 reference		65		dB
		1 MSamples/s, 12 bit, differential, 5V reference		54		dB
		1 MSamples/s, 12 bit, differential, VDD reference		67		dB
		1 MSamples/s, 12 bit, differential, 2xVDD reference		69		dB
		200 kSamples/s, 12 bit, single ended, internal 1v25 reference		62		dB
		200 kSamples/s, 12 bit, single ended, internal 2v5 reference		63		dB
		200 kSamples/s, 12 bit, single ended, VDD reference		67		dB
		200 kSamples/s, 12 bit, differential, internal 1v25 reference		63		dB
		200 kSamples/s, 12 bit, differential, internal 2v5 reference		66		dB
		200 kSamples/s, 12 bit, differential, 5V reference		66		dB
		200 kSamples/s, 12 bit, differential, VDD reference		69		dB
		200 kSamples/s, 12 bit, differential, 2xVDD reference		70		dB
SNDRADC	Signal to Noise-pulse-Distortion Ratio (SNDR)	1 MSamples/s, 12 bit, single ended, internal 1v25 reference		58		dB
		1 MSamples/s, 12 bit, single ended, internal 2v5 reference		62		dB
		1 MSamples/s, 12 bit, single ended, VDD reference		64		dB
		1 MSamples/s, 12 bit, differential, internal 1v25 reference		60		dB
		1 MSamples/s, 12 bit, differential, internal 2v5 reference		64		dB
		1 MSamples/s, 12 bit, differential, 5V reference		54		dB
		1 MSamples/s, 12 bit, differential, VDD reference		66		dB
		1 MSamples/s, 12 bit, differential, 2xVDD reference		68		dB
		200 kSamples/s, 12 bit, single ended, internal 1v25 reference		61		dB
		200 kSamples/s, 12 bit, single ended, internal 2v5 reference		65		dB
		200 kSamples/s, 12 bit, single ended, VDD reference		66		dB
		200 kSamples/s, 12 bit, differential, internal 1v25 reference		63		dB
		200 kSamples/s, 12 bit, differential, internal 2v5 reference		66		dB
		200 kSamples/s, 12 bit, differential, 5V reference		66		dB
		200 kSamples/s, 12 bit, differential, VDD reference		68		dB
		200 kSamples/s, 12 bit, differential, 2xVDD reference		69		dB
SFDRADC	Spurious-Free Dynamic Range (SFDR)	1 MSamples/s, 12 bit, single ended, internal 1v25 reference		64		dBc
		1 MSamples/s, 12 bit, single ended, internal 2v5 reference		76		dBc
		1 MSamples/s, 12 bit, single ended, VDD reference		73		dBc
		1 MSamples/s, 12 bit, differential, internal 1v25 reference		66		dBc
		1 MSamples/s, 12 bit, differential, internal 2v5 reference		77		dBc
		1 MSamples/s, 12 bit, differential, VDD reference		76		dBc
		1 MSamples/s, 12 bit, differential, 2xVDD reference		75		dBc
		1 MSamples/s, 12 bit, differential, 5V reference		69		dBc
		200 kSamples/s, 12 bit, single ended, internal 1v25 reference		75		dBc
		200 kSamples/s, 12 bit, single ended, internal 2v5 reference		75		dBc
		200 kSamples/s, 12 bit, single ended, VDD reference		76		dBc
		200 kSamples/s, 12 bit, differential, internal 1v25 reference		79		dBc
		200 kSamples/s, 12 bit, differential, internal 2v5 reference		79		dBc
		200 kSamples/s, 12 bit, differential, 5V reference		78		dBc
		200 kSamples/s, 12 bit, differential, VDD reference		79		dBc
		200 kSamples/s, 12 bit, differential, 2xVDD reference		79		dBc

[1] On the average every ADC will have one missing code, most likely to appear around 2048 +/- n*512 where n can be a value in the set -3, -2, -1, 1, 2, 3. There will be no missing code around 2048, and in spite of the missing code the ADC will be monotonic at all times so that a response to a slowly increasing input will always be a slowly increasing output. Around the one code that is missing, the neighbour codes will look wider in the DNL plot. The spectra will show spurs on the level of -78dBc for a full scale input for chips that have the missing code issue.

[2] Typical numbers given by abs(Mean) / (85 - 25).

[3] Max number given by (abs(Mean) + 3x stddev) / (85 - 25).

13.8 DAC

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VDACOUT	Output voltage range	VDD voltage reference, single ended	0		VDD	V
		VDD voltage reference, differential	-VDD		VDD	V
VDACCM	Output common mode voltage range		0		VDD	V
IDAC	Active current including references for 2 channels	500 kSamples/s, 12bit		400		µA
		100 kSamples/s, 12 bit			200	µA
		1 kSamples/s 12 bit NORMAL		38		µA
SRDAC	Sample rate				500	ksamples/s
fDAC	DAC clock frequency	Continuous Mode				1000 kHz
		Sample/Hold Mode			250	kHz
		Sample/Off Mode			250	kHz
CYCDACCONV	Clock cycles per conversion			2		
tDACCONV	Conversion time			2		µs
tDACSETTLE	Settling time			5		µs
SNRDAC	Signal to Noise Ratio (SNR)	500 kSamples/s, 12 bit, single ended, internal 1v25 reference		58		dB
		500 kSamples/s, 12 bit, single ended, internal 2v5 reference		59		dB
		500 kSamples/s, 12 bit, differential, internal 1v25 reference		58		dB
		500 kSamples/s, 12 bit, differential, internal 2v5 reference		58		dB
		500 kSamples/s, 12 bit, differential, VDD reference		59		dB
SNDRDAC	Signal to Noise-pulse Distortion Ratio (SNDR)	500 kSamples/s, 12 bit, single ended, internal 1v25 reference		57		dB
		500 kSamples/s, 12 bit, single ended, internal 2v5 reference		54		dB
		500 kSamples/s, 12 bit, differential, internal 1v25 reference		56		dB
		500 kSamples/s, 12 bit, differential, internal 2v5 reference		53		dB
		500 kSamples/s, 12 bit, differential, VDD reference		55		dB
SFDRDAC	Spurious-Free Dynamic Range (SFDR)	500 kSamples/s, 12 bit, single ended, internal 1v25 reference		62		dBc
		500 kSamples/s, 12 bit, single ended, internal 2v5 reference		56		dBc
		500 kSamples/s, 12 bit, differential, internal 1v25 reference		61		dBc
		500 kSamples/s, 12 bit, differential, internal 2v5 reference		55		dBc
		500 kSamples/s, 12 bit, differential, VDD reference		60		dBc
VDACOFFSET	Offset voltage	After calibration, single ended			2	mV
		After calibration, differential			2	mV
DNLDAC	Differential non-linearity			Â±1		LSB
INLDAC	Integral non-linearity				Â±5	LSB
MCDAC	No missing codes			12		bits

13.9 Operational Amplifier (OPAMP)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
IOPAMP	Active Current	(OPA2)BIASPROG=0xF, (OPA2)HALFBIAS=0x0, Unity Gain		400		µA
		(OPA2)BIASPROG=0x7, (OPA2)HALFBIAS=0x1, Unity Gain		100		µA
		(OPA2)BIASPROG=0x0, (OPA2)HALFBIAS=0x1, Unity Gain		13		µA
GOL	Open Loop Gain	(OPA2)BIASPROG=0xF, (OPA2)HALFBIAS=0x0		101		dB
		(OPA2)BIASPROG=0x7, (OPA2)HALFBIAS=0x1		98		dB
		(OPA2)BIASPROG=0x0, (OPA2)HALFBIAS=0x1		91		dB
GBWOPAMP	Gain Bandwidth Product	(OPA2)BIASPROG=0xF, (OPA2)HALFBIAS=0x0		6.1		MHz
		(OPA2)BIASPROG=0x7, (OPA2)HALFBIAS=0x1		1.8		MHz
		(OPA2)BIASPROG=0x0, (OPA2)HALFBIAS=0x1		0.25		MHz
PMOPAMP	Phase Margin	(OPA2)BIASPROG=0xF, (OPA2)HALFBIAS=0x0, CL=75 pF		64		°
		(OPA2)BIASPROG=0x7, (OPA2)HALFBIAS=0x1, CL=75 pF		58		°
		(OPA2)BIASPROG=0x0, (OPA2)HALFBIAS=0x1, CL=75 pF		58		°
RINPUT	Input Resistance			100		Mohm
RLOAD	Load Resistance			200		Ohm
ILOAD_DC	DC Load Current				11	mA
VINPUT	Input Voltage	OPAxHCMDIS=0	VSS		VDD	V
		OPAxHCMDIS=1	VSS		VDD-1.2	V
VOUTPUT	Output Voltage		VSS		VDD	V
VOFFSET	Input Offset Voltage	Unity Gain, VSS<Vin<DD, OPAxHCMDIS=0		6		mV
		Unity Gain, VSS<Vin<DD-1.2, OPAxHCMDIS=1		1		mV
VOFFSET_DRIFT	Input Offset Voltage Drift				0.02	mV/°C
SROPAMP	Slew Rate	(OPA2)BIASPROG=0xF, (OPA2)HALFBIAS=0x0		3.2		V/us
		(OPA2)BIASPROG=0x7, (OPA2)HALFBIAS=0x1		0.8		V/us
		(OPA2)BIASPROG=0x0, (OPA2)HALFBIAS=0x1		0.1		V/us
NOPAMP	Voltage Noise	Vout=1V, RESSEL=0, 0.1 Hz<f<10 kHz, OPAx-HCMDIS=0		101		µVRMS
		Vout=1V, RESSEL=0, 0.1 Hz<f<10 kHz, OPAx-HCMDIS=1		141		µVRMS
		Vout=1V, RESSEL=0, 0.1 Hz<f<1 MHz, OPAx-HCMDIS=0		196		vVRMS
		Vout=1V, RESSEL=0, 0.1 Hz<f<1 MHz, OPAx-HCMDIS=1		229		µVRMS
		RESSEL=7, 0.1 Hz<f<10 kHz, OPAxHCMDIS=0		1230		µVRMS
		RESSEL=7, 0.1 Hz<f<10 kHz, OPAxHCMDIS=1		2130		µVRMS
		RESSEL=7, 0.1 Hz<f<1 MHz, OPAxHCMDIS=0		1630		µVRMS
		RESSEL=7, 0.1 Hz<f<1 MHz, OPAxHCMDIS=1		2590		µVRMS

13.10 Analog comparator(ACMP)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VACMPIN	Input voltage range		0		VDD	V
VACMPCM	ACMP Common Mode voltage range			0	VDD	V
IACMP	Active current	BIASPROG=0b0000, FULL-BIAS=0 and HALFBIAS=1 in ACMPn_CTRL register		0.1		µA
		BIASPROG=0b1111, FULL-BIAS=0 and HALFBIAS=0 in ACMPn_CTRL register		2.87		µA
		BIASPROG=0b1111, FULL-BIAS=1 and HALFBIAS=0 in ACMPn_CTRL register		195		µA
IACMPREF	Current consumption of internal voltage reference	Internal voltage reference off. Using external voltage reference		0		uA
		Internal voltage reference		5		µA
VACMPOFFSET	Offset voltage	Single ended		10		mV
		Differential		10		mV
VACMPHYST	ACMP hysteresis	Programmable		17		mV
RCSRES	Capacitive Sense Internal Resistance	CSRESSEL=0b00 in ACMPn_INPUTSEL		39		kOhm
		CSRESSEL=0b01 in ACMPn_INPUTSEL		71		kOhm
		CSRESSEL=0b10 in ACMPn_INPUTSEL		104		kOhm
		CSRESSEL=0b11 in ACMPn_INPUTSEL		136		kOhm

The total ACMP current is the sum of the contributions from the ACMP and its internal voltage reference as shown below. IACMPREF is zero if an external voltage reference is used.

$$IACMPTOTAL = IACMP + IACMPREF \quad (1)$$

13.11 Voltage comparator(VCMP)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VVCMPIN	Input voltage range			VDD		V
VVCMPCM	VCMP Common Mode voltage range			VDD		V
IVCMP	Active current	BIASPROG=0b0000 and HALFBIAS=1 in VCMPn_CTRL register		0.1		µA
		BIASPROG=0b1111 and HALFBIAS=0 in VCMPn_CTRL register. LPREF=0.		14.7		µA
tVCMPREF	Startup time reference generator	NORMAL		10		µs
VVCMPOFFSET	Offset voltage	Single ended		10		mV
		Differential		10		mV
VVCMPHYST	VCMP hysteresis			17		mV

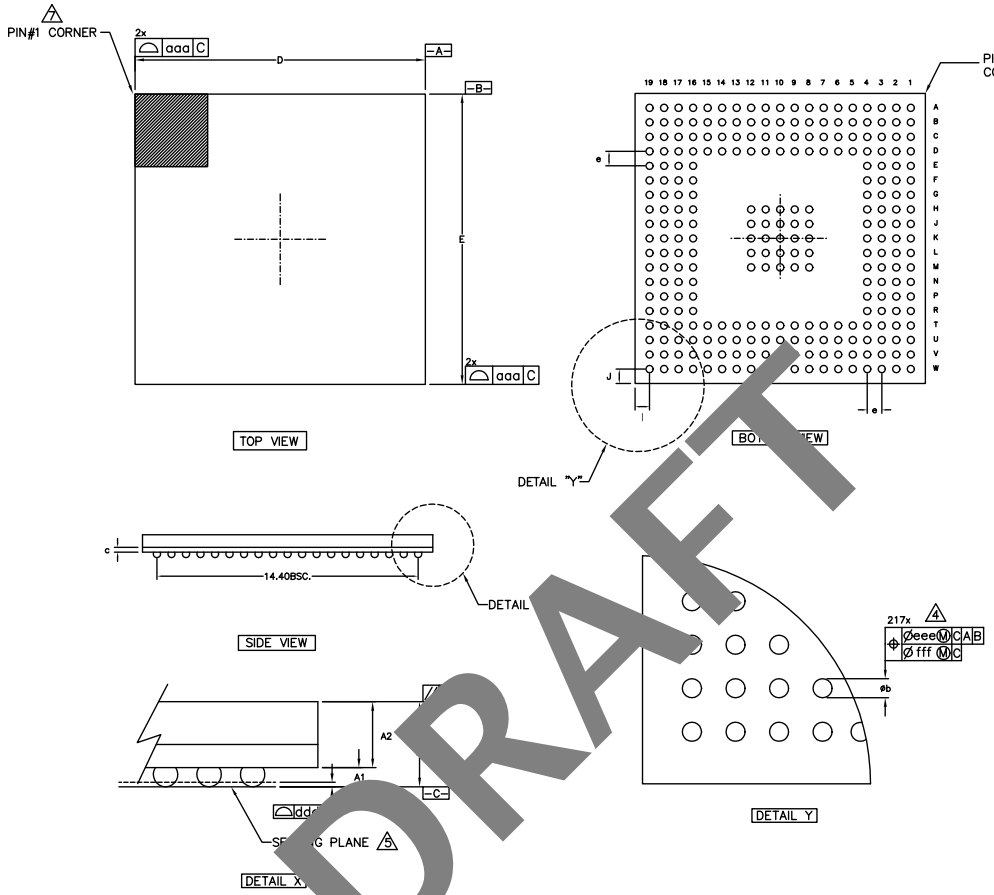
The VDD trigger level can be configured by setting the TRIGLEVEL field of the VCMP_CTRL register in accordance with the following equation:

$$\text{VDD Trigger Level} = 1.667V + 0.034 \times \text{TRIGLEVEL} \quad (2)$$

13.12 Digital Peripherals

Symbol	Parameter	Condition	Min	Typ	Max	Unit
IUSART	USART current	USART idle current, clock enabled		7.5		µA/ MHz
IUART	UART current	UART idle current, clock enabled		5.63		µA/ MHz
ILEUART	LEUART current	LEUART idle current, clock enabled		150		nA
II2C	I2C current	I2C idle current, clock enabled		6.25		µA/ MHz
ITIMER	TIMER current	TIMER_0 idle current, clock enabled		8.75		µA/ MHz
ILETIMER	LETIMER current	LETIMER idle current, clock enabled		150		nA
IPCNT	PCNT current	PCNT idle current, clock enabled		100		nA
IRTC	RTC current	RTC idle current, clock enabled		100		nA
IAES	AES current	AES idle current, clock enabled		2.5		µA/ MHz
IGPIO	GPIO current	GPIO idle current, clock enabled		5.31		µA/ MHz
IEBI	EBI current	EBI idle current, clock enabled		1.56		µA/ MHz
IPRS	PRS current	PRS idle current		2.81		µA/ MHz
IDMA	DMA current	Clock enable		8.12		µA/ MHz

14 Package Information



NOTE:

1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. "e" REPRESENTS THE BASIC SOLDER BALL GRID PITCH.
3. "M" REPRESENTS THE MAXIMUM SOLDER BALL MATRIX SIZE.
4. DEMENSIONS "b" IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER PARALLEL TO PRIMARY DATUM -C- .
5. PRIMARY DATUM -C- AND SEATING PLANE ARE DESIGNED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
6. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994
7. A1 CORNER MUST BE IDENTIFIED BY INK OR LASER MARK.
8. PACKAGE DIMENSIONS CONFORM TO JEDEC REGISTRATION MO–275.

SYMBOL	MIN.	NOM.	MAX.
A	1.16	1.26	1.36
A1	0.25	0.30	0.35
A2	0.91	0.96	1.01
D	15.90	16.00	16.10
E	15.90	16.00	16.10
I		0.80 REF.	
J		0.80 REF.	
M		19 X 19<DEPOPULATED>	
aaa			0.15
ccc			0.20
ddd			0.10
eee			0.15
fff			0.08
b	0.35	0.40	0.45
e		0.80 BSC.	
c		0.26 REF.	

14.1 Part Marking

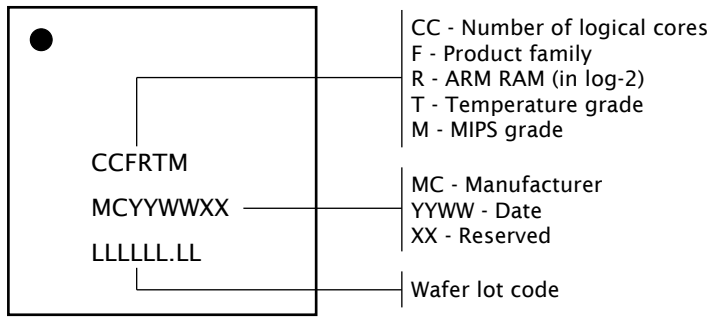


Figure 32: Part marking scheme

15 Ordering Information

Figure 33: Orderable part numbers

Product Code	Marking	Qualification	Speed Grade
XS1-XAU8A-10-FB265-C5	8XAU10C5	Commercial	500 MIPS
XS1-XAU8A-10-FB265-I5	8XAU10I5	Industrial	500 MIPS

Appendices

A Configuration of the xCORE Tile

The xCORE tile is configured through three banks of registers, as shown in Figure 34.

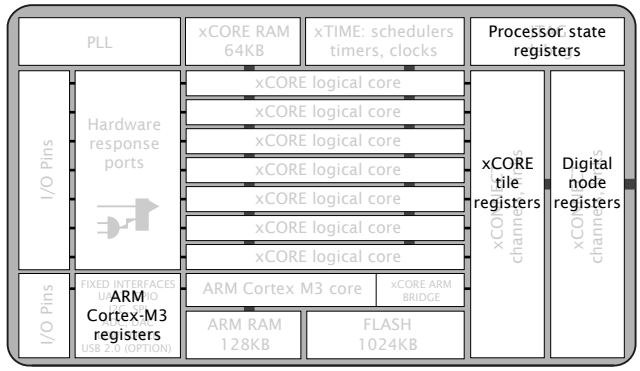


Figure 34: Registers

The following communication sequences specify how to access those registers. Any messages transmitted contain the most significant 24 bits of the channel-end to which a response is to be sent. This comprises the node-identifier and the channel number within the node. If no response is required on a write operation, supply 24-bits with the last 8-bits set, which suppresses the reply message. Any multi-byte data is sent most significant byte first.

A.1 Accessing a processor status register

The processor status registers are accessed directly from the processor instruction set. The instructions GETPS and SETPS read and write a word. The register number should be translated into a processor-status resource identifier by shifting the register number left 8 places, and ORing it with 0x0C. Alternatively, the functions `getps(reg)` and `setps(reg,value)` can be used from XC.

A.2 Accessing an xCORE Tile configuration register

xCORE Tile configuration registers can be accessed through the interconnect using the functions `write_tile_config_reg(tileref, ...)` and `read_tile_config_reg(tile ↪ ref, ...)`, where `tileref` is the name of the xCORE Tile, e.g. `tile[1]`. These functions implement the protocols described below.

Instead of using the functions above, a channel-end can be allocated to communicate with the xCORE tile configuration registers. The destination of the channel-end should be set to `0xnnnnC20C` where `nnnn` is the tile-identifier.

A write message comprises the following:

control-token 192	24-bit response channel-end identifier	16-bit register number	32-bit data	control-token 1

The response to a write message comprises either control tokens 3 and 1 (for success), or control tokens 4 and 1 (for failure).

A read message comprises the following:

control-token 193	24-bit response channel-end identifier	16-bit register number	control-token 1

The response to the read message comprises either control token 3, 32-bit of data, and control-token 1 (for success), or control tokens 4 and 1 (for failure).

A.3 Accessing node configuration

Node configuration registers can be accessed through the interconnect using the functions `write_node_config_reg(device, ...)` and `read_node_config_reg(device, ...)`, where `device` is the name of the node. These functions implement the protocols described below.

Instead of using the functions above, a channel-end can be allocated to communicate with the node configuration registers. The destination of the channel-end should be set to `0xnnnnC30C` where `nnnn` is the node-identifier.

A write message comprises the following:

control-token 192	24-bit response channel-end identifier	16-bit register number	32-bit data	control-token 1

The response to a write message comprises either control tokens 3 and 1 (for success), or control tokens 4 and 1 (for failure).

A read message comprises the following:

control-token 193	24-bit response channel-end identifier	16-bit register number	control-token 1

The response to a read message comprises either control token 3, 32-bit of data, and control-token 1 (for success), or control tokens 4 and 1 (for failure).

B Processor Status Configuration

The processor status control registers can be accessed directly by the processor using processor status reads and writes (use `getps(reg)` and `setps(reg,value)` for reads and writes).

Number	Perm	Description
0x00	RW	RAM base address
0x01	RW	Vector base address
0x02	RW	xCORE Tile control
0x03	RO	xCORE Tile boot status
0x05	RO	Security configuration
0x06	RW	Ring Oscillator Control
0x07	RO	Ring Oscillator Value
0x08	RO	Ring Oscillator Value
0x09	RO	Ring Oscillator Value
0x0A	RO	Ring Oscillator Value
0x10	DRW	Debug SSR
0x11	DRW	Debug SPC
0x12	DRW	Debug SSP
0x13	DRW	DGETREG operand 1
0x14	DRW	DGETREG operand 2
0x15	DRW	Debug interrupt type
0x16	DRW	Debug interrupt data
0x18	DRW	Debug core control
0x20 .. 0x27	DRW	Debug scratch
0x30 .. 0x33	DRW	Instruction breakpoint address
0x40 .. 0x43	DRW	Instruction breakpoint control
0x50 .. 0x53	DRW	Data watchpoint address 1
0x60 .. 0x63	DRW	Data watchpoint address 2
0x70 .. 0x73	DRW	Data breakpoint control register
0x80 .. 0x83	DRW	Resources breakpoint mask
0x90 .. 0x93	DRW	Resources breakpoint value
0x9C .. 0x9F	DRW	Resources breakpoint control register

Figure 35: Summary

B.1 RAM base address: 0x00

This register contains the base address of the RAM. It is initialized to 0x00010000.

0x00: RAM base address

Bits	Perm	Init	Description
31:2	RW		Most significant 16 bits of all addresses.
1:0	RO	-	Reserved

B.2 Vector base address: 0x01

Base address of event vectors in each resource. On an interrupt or event, the 16 most significant bits of the destination address are provided by this register; the least significant 16 bits come from the event vector.

0x01: Vector base address

Bits	Perm	Init	Description
31:16	RW		The most significant bits for all event and interrupt vectors.
15:0	RO	-	Reserved

B.3 xCORE Tile control: 0x02

Register to control features in the xCORE tile

0x02: xCORE Tile control

Bits	Perm	Init	Description
31:6	RO	-	Reserved
5	RW	0	Set to 1 to select the dynamic mode for the clock divider when the clock divider is enabled. In dynamic mode the clock divider is only activated when all active logical cores are paused. In static mode the clock divider is always enabled.
4	RW	0	Set to 1 to enable the clock divider. This slows down the xCORE tile clock in order to use less power.
3:0	RO	-	Reserved

B.4 xCORE Tile boot status: 0x03

This read-only register describes the boot status of the xCORE tile.

0x03: xCORE Tile boot status

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	RO		xCORE tile number on the switch.
15:9	RO	-	Reserved
8	RO		Set to 1 if boot from OTP is enabled.
7:0	RO		The boot mode pins MODE0, MODE1, ..., specifying the boot frequency, boot source, etc.

B.5 Security configuration: 0x05

Copy of the security register as read from OTP.

0x05: Security configuration

Bits	Perm	Init	Description
31:0	RO		Value.

B.6 Ring Oscillator Control: 0x06

There are four free-running oscillators that clock four counters. The oscillators can be started and stopped using this register. The counters should only be read when the ring oscillator is stopped. The counter values can be read using four subsequent registers. The ring oscillators are asynchronous to the xCORE tile clock and can be used as a source of random bits.

0x06: Ring Oscillator Control

Bits	Perm	Init	Description
31:2	RO	-	Reserved
1	RW	0	Set to 1 to enable the xCORE tile ring oscillators
0	RW	0	Set to 1 to enable the peripheral ring oscillators

B.7 Ring Oscillator Value: 0x07

This register contains the current count of the xCORE Tile Cell ring oscillator. This value is not reset on a system reset.

0x07: Ring Oscillator Value

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RO	-	Ring oscillator counter data.

B.8 Ring Oscillator Value: 0x08

This register contains the current count of the xCORE Tile Wire ring oscillator. This value is not reset on a system reset.

0x08: Ring Oscillator Value

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RO	-	Ring oscillator counter data.

B.9 Ring Oscillator Value: 0x09

This register contains the current count of the Peripheral Cell ring oscillator. This value is not reset on a system reset.

0x09: Ring Oscillator Value

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RO	-	Ring oscillator counter data.

B.10 Ring Oscillator Value: 0x0A

This register contains the current count of the Peripheral Wire ring oscillator. This value is not reset on a system reset.

0x0A: Ring Oscillator Value

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RO	-	Ring oscillator counter data.

B.11 Debug SSR: 0x10

This register contains the value of the SSR register when the debugger was called.

0x10: Debug SSR

Bits	Perm	Init	Description
31:0	RO	-	Reserved

B.12 Debug SPC: 0x11

This register contains the value of the SPC register when the debugger was called.

0x11: Debug SPC

Bits	Perm	Init	Description
31:0	DRW		Value.

B.13 Debug SSP: 0x12

This register contains the value of the SSP register when the debugger was called.

0x12: Debug SSP

Bits	Perm	Init	Description
31:0	DRW		Value.

B.14 DGETREG operand 1: 0x13

The resource ID of the logical core whose state is to be read.

0x13: DGETREG operand 1

Bits	Perm	Init	Description
31:8	RO	-	Reserved
7:0	DRW		Thread number to be read

B.15 DGETREG operand 2: 0x14

Register number to be read by DGETREG

0x14: DGETREG operand 2

Bits	Perm	Init	Description
31:5	RO	-	Reserved
4:0	DRW		Register number to be read

B.16 Debug interrupt type: 0x15

Register that specifies what activated the debug interrupt.

0x15: Debug interrupt type

Bits	Perm	Init	Description
31:18	RO	-	Reserved
17:16	DRW		If the debug interrupt was caused by a hardware breakpoint or hardware watchpoint, this field contains the number of the breakpoint or watchpoint. If multiple breakpoints or watchpoints trigger at once, the lowest number is taken.
15:8	DRW		If the debug interrupt was caused by a logical core, this field contains the number of that core. Otherwise this field is 0.
7:3	RO	-	Reserved
2:0	DRW	0	Indicates the cause of the debug interrupt 1: Host initiated a debug interrupt through JTAG 2: Program executed a DCALL instruction 3: Instruction breakpoint 4: Data watch point 5: Resource watch point

B.17 Debug interrupt data: 0x16

On a data watchpoint, this register contains the effective address of the memory operation that triggered the debugger. On a resource watchpoint, it countains the resource identifier.

0x16: Debug interrupt data

Bits	Perm	Init	Description
31:0	DRW		Value.

B.18 Debug core control: 0x18

This register enables the debugger to temporarily disable logical cores. When returning from the debug interrupts, the cores set in this register will not execute. This enables single stepping to be implemented.

0x18: Debug core control

Bits	Perm	Init	Description
31:8	RO	-	Reserved
7:0	DRW		1-hot vector defining which logical cores are stopped when not in debug mode. Every bit which is set prevents the respective logical core from running.

B.19 Debug scratch: 0x20 .. 0x27

A set of registers used by the debug ROM to communicate with an external debugger, for example over JTAG. This is the same set of registers as the Debug Scratch registers in the xCORE tile configuration.

0x20 .. 0x27: Debug scratch

Bits	Perm	Init	Description
31:0	DRW		Value.

B.20 Instruction breakpoint address: 0x30 .. 0x33

This register contains the address of the instruction breakpoint. If the PC matches this address, then a debug interrupt will be taken. There are four instruction breakpoints that are controlled individually.

0x30 .. 0x33: Instruction breakpoint address

Bits	Perm	Init	Description
31:0	DRW		Value.

B.21 Instruction breakpoint control: 0x40 .. 0x43

This register controls which logical cores may take an instruction breakpoint, and under which condition.

0x40 .. 0x43: Instruction breakpoint control

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	DRW	0	A bit for each logical core in the tile allowing the breakpoint to be enabled individually for each logical core.
15:2	RO	-	Reserved
1	DRW	0	Set to 1 to cause an instruction breakpoint if the PC is not equal to the breakpoint address. By default, the breakpoint is triggered when the PC is equal to the breakpoint address.
0	DRW	0	When 1 the instruction breakpoint is enabled.

B.22 Data watchpoint address 1: 0x50 .. 0x53

This set of registers contains the first address for the four data watchpoints.

0x50 .. 0x53: Data watchpoint address 1

Bits	Perm	Init	Description
31:0	DRW		Value.

B.23 Data watchpoint address 2: 0x60 .. 0x63

This set of registers contains the second address for the four data watchpoints.

0x60 .. 0x63: Data watchpoint address 2

Bits	Perm	Init	Description
31:0	DRW		Value.

B.24 Data breakpoint control register: 0x70 .. 0x73

This set of registers controls each of the four data watchpoints.

0x70 .. 0x73: Data breakpoint control register

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	DRW	0	A bit for each logical core in the tile allowing the breakpoint to be enabled individually for each logical core.
15:3	RO	-	Reserved
2	DRW	0	Set to 1 to enable breakpoints to be triggered on loads. Breakpoints always trigger on stores.
1	DRW	0	By default, data watchpoints trigger if memory in the range [Address1..Address2] is accessed (the range is inclusive of Address1 and Address2). If set to 1, data watchpoints trigger if memory outside the range (Address2..Address1) is accessed (the range is exclusive of Address2 and Address1).
0	DRW	0	When 1 the instruction breakpoint is enabled.

B.25 Resources breakpoint mask: 0x80 .. 0x83

This set of registers contains the mask for the four resource watchpoints.

0x80 .. 0x83: Resources breakpoint mask

Bits	Perm	Init	Description
31:0	DRW		Value.

B.26 Resources breakpoint value: 0x90 .. 0x93

This set of registers contains the value for the four resource watchpoints.

0x90 .. 0x93: Resources breakpoint value

Bits	Perm	Init	Description
31:0	DRW		Value.

B.27 Resources breakpoint control register: 0x9C .. 0x9F

This set of registers controls each of the four resource watchpoints.

0x9C .. 0x9F: Resources breakpoint control register

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	DRW	0	A bit for each logical core in the tile allowing the breakpoint to be enabled individually for each logical core.
15:2	RO	-	Reserved
1	DRW	0	By default, resource watchpoints trigger when the resource id masked with the set Mask equals the Value. If set to 1, resource watchpoints trigger when the resource id masked with the set Mask is not equal to the Value.
0	DRW	0	When 1 the instruction breakpoint is enabled.

C Tile Configuration

The xCORE Tile control registers can be accessed using configuration reads and writes (use `write_tile_config_reg(tileref, ...)` and `read_tile_config_reg(tileref, ...)` for reads and writes).

Number	Perm	Description
0x00	RO	Device identification
0x01	RO	xCORE Tile description 1
0x02	RO	xCORE Tile description 2
0x04	CRW	Control PSwitch permissions to debug registers
0x05	CRW	Cause debug interrupts
0x06	RW	xCORE Tile clock divider
0x07	RO	Security configuration
0x10 .. 0x13	RO	PLink status
0x20 .. 0x27	CRW	Debug scratch
0x40	RO	PC of logical core 0
0x41	RO	PC of logical core 1
0x42	RO	PC of logical core 2
0x43	RO	PC of logical core 3
0x44	RO	PC of logical core 4
0x45	RO	PC of logical core 5
0x46	RO	PC of logical core 6
0x47	RO	PC of logical core 7
0x60	RO	SR of logical core 0
0x61	RO	SR of logical core 1
0x62	RO	SR of logical core 2
0x63	RO	SR of logical core 3
0x64	RO	SR of logical core 4
0x65	RO	SR of logical core 5
0x66	RO	SR of logical core 6
0x67	RO	SR of logical core 7
0x80 .. 0x9F	RO	Chanend status

Figure 36: Summary

C.1 Device identification: 0x00

0x00: Device identification

Bits	Perm	Init	Description
31:24	RO		Processor ID of this xCORE tile.
23:16	RO		Number of the node in which this xCORE tile is located.
15:8	RO		xCORE tile revision.
7:0	RO		xCORE tile version.

C.2 xCORE Tile description 1: 0x01

This register describes the number of logical cores, synchronisers, locks and channel ends available on this xCORE tile.

0x01: xCORE Tile description 1

Bits	Perm	Init	Description
31:24	RO		Number of channel ends.
23:16	RO		Number of locks.
15:8	RO		Number of synchronisers.
7:0	RO	-	Reserved

C.3 xCORE Tile description 2: 0x02

This register describes the number of timers and clock blocks available on this xCORE tile.

0x02: xCORE Tile description 2

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:8	RO		Number of clock blocks.
7:0	RO		Number of timers.

C.4 Control PSwitch permissions to debug registers: 0x04

This register can be used to control whether the debug registers (marked with permission CRW) are accessible through the tile configuration registers. When this bit is set, write -access to those registers is disabled, preventing debugging of the xCORE tile over the interconnect.

0x04: Control PSwitch permissions to debug registers

Bits	Perm	Init	Description
31:1	RO	-	Reserved
0	CRW		Set to 1 to restrict PSwitch access to all CRW marked registers to become read-only rather than read-write.

C.5 Cause debug interrupts: 0x05

This register can be used to raise a debug interrupt in this xCORE tile.

0x05: Cause debug interrupts

Bits	Perm	Init	Description
31:2	RO	-	Reserved
1	RO	0	Set to 1 when the processor is in debug mode.
0	CRW	0	Set to 1 to request a debug interrupt on the processor.

C.6 xCORE Tile clock divider: 0x06

This register contains the value used to divide the PLL clock to create the xCORE tile clock. The divider is enabled under control of the tile control register

0x06: xCORE Tile clock divider

Bits	Perm	Init	Description
31:8	RO	-	Reserved
7:0	RW		Value of the clock divider minus one.

C.7 Security configuration: 0x07

Copy of the security register as read from OTP.

0x07: Security configuration

Bits	Perm	Init	Description
31:0	RO		Value.

C.8 PLink status: 0x10 .. 0x13

Status of each of the four processor links; connecting the xCORE tile to the switch.

0x10 .. 0x13: PLink status

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		00 - ChannelEnd, 01 - ERROR, 10 - PSCTL, 11 - Idle.
23:16	RO		Based on SRC_TARGET_TYPE value, it represents channelEnd ID or Idle status.
15:6	RO	-	Reserved
5:4	RO		Two-bit network identifier
3	RO	-	Reserved
2	RO		1 when the current packet is considered junk and will be thrown away.
1	RO	0	Set to 1 if the switch is routing data into the link, and if a route exists from another link.
0	RO	0	Set to 1 if the link is routing data into the switch, and if a route is created to another link on the switch.

C.9 Debug scratch: 0x20 .. 0x27

A set of registers used by the debug ROM to communicate with an external debugger, for example over the switch. This is the same set of registers as the Debug Scratch registers in the processor status.

0x20 .. 0x27: Debug scratch

Bits	Perm	Init	Description
31:0	CRW		Value.

C.10 PC of logical core 0: 0x40

Value of the PC of logical core 0.

0x40: PC of logical core 0

Bits	Perm	Init	Description
31:0	RO		Value.

C.11 PC of logical core 1: 0x41

0x41: PC of logical core 1

Bits	Perm	Init	Description
31:0	RO		Value.

C.12 PC of logical core 2: 0x42

0x42: PC of logical core 2

Bits	Perm	Init	Description
31:0	RO		Value.

C.13 PC of logical core 3: 0x43

0x43: PC of logical core 3

Bits	Perm	Init	Description
31:0	RO		Value.

C.14 PC of logical core 4: 0x44

0x44: PC of logical core 4

Bits	Perm	Init	Description
31:0	RO		Value.

C.15 PC of logical core 5: 0x45

0x45: PC of logical core 5

Bits	Perm	Init	Description
31:0	RO		Value.

C.16 PC of logical core 6: 0x46

0x46: PC of logical core 6

Bits	Perm	Init	Description
31:0	RO		Value.

C.17 PC of logical core 7: 0x47

0x47: PC of logical core 7

Bits	Perm	Init	Description
31:0	RO		Value.

C.18 SR of logical core 0: 0x60

Value of the SR of logical core 0

0x60: SR of logical core 0

Bits	Perm	Init	Description
31:0	RO		Value.

C.19 SR of logical core 1: 0x61

0x61: SR of logical core 1

Bits	Perm	Init	Description
31:0	RO		Value.

C.20 SR of logical core 2: 0x62

0x62: SR of logical core 2

Bits	Perm	Init	Description
31:0	RO		Value.

C.21 SR of logical core 3: 0x63

0x63: SR of logical core 3

Bits	Perm	Init	Description
31:0	RO		Value.

C.22 SR of logical core 4: 0x64

0x64: SR of logical core 4

Bits	Perm	Init	Description
31:0	RO		Value.

C.23 SR of logical core 5: 0x65

0x65: SR of logical core 5

Bits	Perm	Init	Description
31:0	RO		Value.

C.24 SR of logical core 6: 0x66

0x66: SR of logical core 6

Bits	Perm	Init	Description
31:0	RO		Value.

C.25 SR of logical core 7: 0x67

0x67: SR of logical core 7

Bits	Perm	Init	Description
31:0	RO		Value.

C.26 Chanend status: 0x80 .. 0x9F

These registers record the status of each channel-end on the tile.

0x80 .. 0x9F: Chanend status

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		00 - ChannelEnd, 01 - ERROR, 10 - PSCTL, 11 - Idle.
23:16	RO		Based on SRC_TARGET_TYPE value, it represents channelEnd ID or Idle status.
15:6	RO	-	Reserved
5:4	RO		Two-bit network identifier
3	RO	-	Reserved
2	RO		1 when the current packet is considered junk and will be thrown away.
1	RO	0	Set to 1 if the switch is routing data into the link, and if a route exists from another link.
0	RO	0	Set to 1 if the link is routing data into the switch, and if a route is created to another link on the switch.

D Node Configuration

The digital node control registers can be accessed using configuration reads and writes (use `write_node_config_reg(device, ...)` and `read_node_config_reg(device, ...)` for reads and writes).

Number	Perm	Description
0x00	RO	Device identification
0x01	RO	System switch description
0x04	RW	Switch configuration
0x05	RW	Switch node identifier
0x06	RW	PLL settings
0x07	RW	System switch clock divider
0x08	RW	Reference clock
0x0C	RW	Directions 0-7
0x0D	RW	Directions 8-15
0x10	RW	DEBUG_N configuration
0x1F	RO	Debug source
0x20 .. 0x27	RW	Link status, direction, and network
0x40 .. 0x43	RW	PLink status and network
0x80 .. 0x87	RW	Link configuration and initialization
0xA0 .. 0xA7	RW	Static link configuration

Figure 37: Summary

D.1 Device identification: 0x00

This register contains version and revision identifiers and the mode-pins as sampled at boot-time.

Bits	Perm	Init	Description
31:24	RO	0x00	Chip identifier.
23:16	RO		Sampled values of pins MODE0, MODE1, ... on reset.
15:8	RO		SSwitch revision.
7:0	RO		SSwitch version.

0x00: Device identification

D.2 System switch description: 0x01

This register specifies the number of processors and links that are connected to this switch.

0x01: System switch description

Bits	Perm	Init	Description
31:24	RO	-	Reserved
23:16	RO		Number of links on the switch.
15:8	RO		Number of cores that are connected to this switch.
7:0	RO		Number of links per processor.

D.3 Switch configuration: 0x04

This register enables the setting of two security modes (that disable updates to the PLL or any other registers) and the header-mode.

0x04: Switch configuration

Bits	Perm	Init	Description
31	RO	0	Set to 1 to disable any write access to the configuration registers in this switch.
30:9	RO	-	Reserved
8	RO	0	Set to 1 to disable updates to the PLL configuration register.
7:1	RO	-	Reserved
0	RO	0	Header mode. Set to 1 to enable 1-byte headers. This must be performed on all nodes in the system.

D.4 Switch node identifier: 0x05

This register contains the node identifier.

0x05: Switch node identifier

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RW	0	The unique 16-bit ID of this node. This ID is matched most-significant-bit first with incoming messages for routing purposes.

D.5 PLL settings: 0x06

An on-chip PLL multiplies the input clock up to a higher frequency clock, used to clock the I/O, processor, and switch, see Oscillator. Note: a write to this register will cause the tile to be reset.

0x06: PLL settings

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:23	RW		OD: Output divider value The initial value depends on pins MODE0 and MODE1.
22:21	RO	-	Reserved
20:8	RW		F: Feedback multiplication ratio The initial value depends on pins MODE0 and MODE1.
7	RO	-	Reserved
6:0	RW		R: Oscilator input divider value The initial value depends on pins MODE0 and MODE1.

D.6 System switch clock divider: 0x07

Sets the ratio of the PLL clock and the switch clock.

0x07: System switch clock divider

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RW	0	Switch clock divider. The PLL clock will be divided by this value plus one to derive the switch clock.

D.7 Reference clock: 0x08

Sets the ratio of the PLL clock and the reference clock used by the node.

0x08: Reference clock

Bits	Perm	Init	Description
31:16	RO	-	Reserved
15:0	RW	3	Architecture reference clock divider. The PLL clock will be divided by this value plus one to derive the 100 MHz reference clock.

D.8 Directions 0-7: 0x0C

This register contains eight directions, for packets with a mismatch in bits 7..0 of the node-identifier. The direction in which a packet will be routed is goverened by the most significant mismatching bit.

0x0C: Directions 0-7

Bits	Perm	Init	Description
31:28	RW	0	The direction for packets whose first mismatching bit is 7.
27:24	RW	0	The direction for packets whose first mismatching bit is 6.
23:20	RW	0	The direction for packets whose first mismatching bit is 5.
19:16	RW	0	The direction for packets whose first mismatching bit is 4.
15:12	RW	0	The direction for packets whose first mismatching bit is 3.
11:8	RW	0	The direction for packets whose first mismatching bit is 2.
7:4	RW	0	The direction for packets whose first mismatching bit is 1.
3:0	RW	0	The direction for packets whose first mismatching bit is 0.

D.9 Directions 8-15: 0x0D

This register contains eight directions, for packets with a mismatch in bits 15..8 of the node-identifier. The direction in which a packet will be routed is goverened by the most significant mismatching bit.

0x0D: Directions 8-15

Bits	Perm	Init	Description
31:28	RW	0	The direction for packets whose first mismatching bit is 15.
27:24	RW	0	The direction for packets whose first mismatching bit is 14.
23:20	RW	0	The direction for packets whose first mismatching bit is 13.
19:16	RW	0	The direction for packets whose first mismatching bit is 12.
15:12	RW	0	The direction for packets whose first mismatching bit is 11.
11:8	RW	0	The direction for packets whose first mismatching bit is 10.
7:4	RW	0	The direction for packets whose first mismatching bit is 9.
3:0	RW	0	The direction for packets whose first mismatching bit is 8.

D.10 DEBUG_N configuration: 0x10

Configures the behavior of the DEBUG_N pin.

0x10: DEBUG_N configuration

Bits	Perm	Init	Description
31:2	RO	-	Reserved
1	RW	0	Set to 1 to enable signals on DEBUG_N to generate DCALL on the core.
0	RW	0	When set to 1, the DEBUG_N wire will be pulled down when the node enters debug mode.

D.11 Debug source: 0x1F

Contains the source of the most recent debug event.

0x1F: Debug source

Bits	Perm	Init	Description
31:5	RO	-	Reserved
4	RW		If set, the external DEBUG_N pin is the source of the most recent debug interrupt.
3:1	RO	-	Reserved
0	RW		If set, the xCORE Tile is the source of the most recent debug interrupt.

D.12 Link status, direction, and network: 0x20 .. 0x27

These registers contain status information for low level debugging (read-only), the network number that each link belongs to, and the direction that each link is part of. The registers control links C, D, A, B, G, H, E, and F in that order.

0x20 .. 0x27: Link status, direction, and network

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		If this link is currently routing data into the switch, this field specifies the type of link that the data is routed to: 0: plink 1: external link 2: internal control link
23:16	RO	0	If the link is routing data into the switch, this field specifies the destination link number to which all tokens are sent.
15:12	RO	-	Reserved
11:8	RW	0	The direction that this this link is associated with; set for routing.
7:6	RO	-	Reserved
5:4	RW	0	Determines the network to which this link belongs, set for quality of service.
3	RO	-	Reserved
2	RO	0	Set to 1 if the current packet is junk and being thrown away. A packet is considered junk if, for example, it is not routable.
1	RO	0	Set to 1 if the switch is routing data into the link, and if a route exists from another link.
0	RO	0	Set to 1 if the link is routing data into the switch, and if a route is created to another link on the switch.

D.13 PLink status and network: 0x40 .. 0x43

These registers contain status information and the network number that each processor-link belongs to.

0x40 .. 0x43: PLink status and network

Bits	Perm	Init	Description
31:26	RO	-	Reserved
25:24	RO		If this link is currently routing data into the switch, this field specifies the type of link that the data is routed to: 0: plink 1: external link 2: internal control link
23:16	RO	0	If the link is routing data into the switch, this field specifies the destination link number to which all tokens are sent.
15:6	RO	-	Reserved
5:4	RW	0	Determines the network to which this link belongs, set for quality of service.
3	RO	-	Reserved
2	RO	0	Set to 1 if the current packet is junk and being thrown away. A packet is considered junk if, for example, it is not routable.
1	RO	0	Set to 1 if the switch is routing data into the link, and if a route exists from another link.
0	RO	0	Set to 1 if the link is routing data into the switch, and if a route is created to another link on the switch.

D.14 Link configuration and initialization: 0x80 .. 0x87

These registers contain configuration and debugging information specific to external links. The link speed and width can be set, the link can be initialized, and the link status can be monitored. The registers control links C, D, A, B, G, H, E, and F in that order.

Bits	Perm	Init	Description
31	RW	0	Write '1' to this bit to enable the link, write '0' to disable it. This bit controls the muxing of ports with overlapping links.
30	RW	0	Set to 0 to operate in 2 wire mode or 1 to operate in 5 wire mode
29:28	RO	-	Reserved
27	RO	0	Set to 1 on error: an RX buffer overflow or illegal token encoding has been received. This bit clears on reading.
26	RO	0	1 if this end of the link has issued credit to allow the remote end to transmit.
25	RO	0	1 if this end of the link has credits to allow it to transmit.
24	WO	0	Set to 1 to initialize a half-duplex link. This clears this end of the link's credit and issues a HELLO token; the other side of the link will reply with credits. This bit is self-clearing.
23	WO	0	Set to 1 to reset the receiver. The next symbol that is detected will be assumed to be the first symbol in a token. This bit is self-clearing.
22	RO	-	Reserved
21:11	RW	0	The number of system clocks between two subsequent transitions within a token
10:0	RW	0	The number of system clocks between two subsequent transmit tokens.

0x80 .. 0x87: Link configuration and initialization

D.15 Static link configuration: 0xA0 .. 0xA7

These registers are used for static (ie, non-routed) links. When a link is made static, all traffic is forwarded to the designated channel end and no routing is attempted. The registers control links C, D, A, B, G, H, E, and F in that order.

Bits	Perm	Init	Description
31	RW	0	Enable static forwarding.
30:5	RO	-	Reserved
4:0	RW	0	The destination channel end on this node that packets received in static mode are forwarded to.

0xA0 .. 0xA7: Static link configuration

E Configuring the ARM core

The ARM-core consists of several different types of peripherals (CMU, RTC, ADCn...). Some peripherals exist only as one instance, like the Clock Management Unit (CMU). Other peripherals like Timers (TIMERn) have several instances - the name is postfixed by a number (n) denoting the instance number. Usually, two instances of a peripheral are identical, but are placed in different regions of the memory map. However, some peripherals have a different feature set for each of the instances, for example USART0 can have an IrDA interface, while USART1 has not.

The peripheral instances each have a dedicated address region which contains registers that can be accessed by read/write operations. The peripheral instances and memory regions are shown in XXX. The starting address of a peripheral instance is called the base address. The address for each register is given as an offset from the base address for the peripheral instance.

The ARM-core uses a 32-bit bus for write/read access to the peripherals, and each register in a peripheral contains 32 bits, numbered 0-31. Unused bits are marked as reserved and should not be modified. The bits used by the peripheral can either be single bits or grouped together in bitfields.

Each register has a set access type for all of the bit fields within that register. The access types describes the reaction to read or write operation to the bit field.

Features supported by the ARM-core are shown below:

Module	Configuration	Pin Connections
Cortex-M3	Full configuration	NA
DBG	Full configuration	DBG_SWCLK, DBG_SWDIO, DBG_SWO
MSC	Full configuration	NA
DMA	Full configuration	NA
RMU	Full configuration	NA
EMU	Full configuration	NA
CMU	Full configuration	CMU_OUT0, CMU_OUT1
WDOG	Full configuration	NA
PRS	Full configuration	NA
USB	Full configuration	USB_VBUS, USB_VBUSEN, USB_VREGI, USB_VREGO, USB_DM, USB_DMPU, USB_DP, USB_ID
EBI	Full configuration	EBI_A[27:0], EBI_AD[15:0], EBI_ARDY, EBI_ALE, EBI_BL[1:0], EBI_CS[3:0], EBI_DCLK, EBI_DTEN, EBI_HSNC, EBI_NANDREn, EBI_NANDWEn, EBI_REn, EBI_VSNC, EBI_WEn
I2C0	Full configuration	I2C0_SDA, I2C0_SCL
I2C1	Full configuration	I2C1_SDA, I2C1_SCL

Continued on next page

Module	Configuration	Pin Connections
USART0	Full configuration with IrDA	US0_TX, US0_RX. US0_CLK, US0_CS
USART1	Full configuration with I2S	US1_TX, US1_RX, US1_CLK, US1_CS
USART2	Full configuration with I2S	US2_TX, US2_RX, US2_CLK, US2_CS
UART0	Full configuration	U0_TX, U0_RX
UART1	Full configuration	U1_TX, U1_RX
LEUART0	Full configuration	LEU0_TX, LEU0_RX
LEUART1	Full configuration	LEU1_TX, LEU1_RX
TIMER0	Full configuration with DTI	TIM0_CC[2:0], TIM0_CDTI[2:0]
TIMER1	Full configuration	TIM1_CC[2:0]
TIMER2	Full configuration	TIM2_CC[2:0]
TIMER3	Full configuration	TIM3_CC[2:0]
RTC	Full configuration	NA
BURTC	Full configuration	NA
LETIMER0	Full configuration	LET0_O[1:0]
PCNT0	Full configuration: 16-bit count register	PCNT0_S[1:0]
PCNT1	Full configuration: 8-bit count register	PCNT1_S[1:0]
PCNT2	Full configuration: 8-bit count register	PCNT2_S[1:0]
ACMP0	Full configuration	ACMP0_CH[7:0], ACMP0_O
ACMP1	Full configuration	ACMP1_CH[7:0], ACMP1_O
VCMP	Full configuration	NA
ADC0	Full configuration	ADC0_CH[7:0]
DAC0	Full configuration	DAC0_OUT[1:0], DAC0_OUTxALT
OPAMP	Full configuration	Outputs: OPAMP_OUTx, OPAMP_OUTxALT, Inputs: OPAMP_Px, OPAMP_Nx
AES	Full configuration	NA
GPIO	70 pins	Available pins are shown in Table X

E.1 ARM-core pinout

The ARM-core pinout is shown below. Alternate locations are denoted by # followed by the location number (Multiple locations on the same pin are split with /). Alternate locations can be configured in the LOCATION bitfield in the *_ROUTE register in the module in question. Pins marked ** are not available if the xCORE-ARM bridge is active.

Pin Name	Analog	EBI	Timers	Communication	Other
PA0**		EBI_AD09_0/1/2	TIM0_CC0_0/1/4	LEU0_RX_4 I2C0_SDA_0	PRS_CH0_0 GPIO_EM4WU0
PA1**		EBI_AD10_0/1/2	TIM0_CC1_0/1	I2C0_SCL_0	CMU_CLK1_0 PRS_CH1_0
PA2**		EBI_AD11_0/1/2	TIM0_CC2_0/1		CMU_CLK0_0 ETM_TD0_3
PA3**		EBI_AD12_0/1/2	TIM0_CDTI0_0	U0_TX_2	LES_ALTEX2_0 ETM_TD1_3
PA4**		EBI_AD13_0/1/2	TIM0_CDTI1_0	U0_RX_2	LES_ALTEX3_0 ETM_TD2_3
PA5**		EBI_AD14_0/1/2	TIM0_CDTI2_0	LEU1_TX_1	LES_ALTEX4_0 ETM_TD3_3
PA6**		EBI_AD15_0/1/2 GPIO_EM4WU1		LEU1_RX_1	ETM_TCLK_3
PA7					
PA8		EBI_DCLK_0/1/2 TIM2_CC0_0			
PA9		EBI_DTEN_0/1/2	TIM2_CC1_0		
PA10		EBI_VSNC_0/1/2 TIM2_CC2_0			
PA11		EBI_HSNC_0/1/2			
PA12		EBI_A00_0/1/2	TIM2_CC0_1		
PA13		EBI_A01_0/1/2 TIM2_CC1_1			
PA14		EBI_A02_0/1/2 TIM2_CC2_1			
PA15**		EBI_AD08_0/1/2	TIM3_CC2_0		
PB0		EBI_A16_0/1/2	TIM1_CC0_2		
PB1		EBI_A17_0/1/2	TIM1_CC1_2		
PB2		EBI_A18_0/1/2	TIM1_CC2_2		
PB3		EBI_A19_0/1/2	PCNT1_S0IN_1	US2_TX_1	
PB4		EBI_A20_0/1/2	PCNT1_S1IN_1	US2_RX_1	
PB5		EBI_A21_0/1/2		US2_CLK_1	
PB6		EBI_A22_0/1/2		US2_CS_1	
PB7	LFXTAL_P		TIM1_CC0_3	US0_TX_4 US1_CLK_0	
PB8	LFXTAL_N		TIM1_CC1_3	US0_RX_4 US1_CS_0	
PB9		EBI_A03_0/1/2		U1_TX_2	
PB10		EBI_A04_0/1/2		U1_RX_2	
PB11	DAC0_OUT0_0 / OPAMP_OUT0		TIM1_CC2_3 LETIM0_OUT0_1	I2C1_SDA_1	
PB12	DAC0_OUT1_0/ OPAMP_OUT1		LETIM0_OUT1_1	I2C1_SCL_1	

Continued on next page

Pin Name	Analog	EBI	Timers	Communication	Other
PB13	HFXTAL_P			USO_CLK_4/5 LEU0_TX_1	
PB15					ETM_TD2_1
PC0	DAC0_OUT0ALT_0/ OPAMP_OUT0ALT ACMP0_CH0	EBI_A23_0/1/2	TIM0_CC1_4 PCNT0_S0IN_2	US0_TX_5 US1_TX_0 I2C0_SDA_4	 LES_CH0_0 PRS_CH2_0
PC1	DAC0_OUT0ALT_1/ OPAMP_OUT0ALT ACMP0_CH1	EBI_A24_0/1/2	TIM0_CC2_4 PCNT0_S1IN_2	US0_RX_5 US1_RX_0 I2C0_SCL_4	LES_CH1_0 PRS_CH3_0
PC2	DAC0_OUT0ALT_2/ OPAMP_OUT0ALT ACMP0_CH2	EBI_A25_0/1/2	TIM0_CDTI0_4	US2_TX_0	LES_CH2_0
PC3	DAC0_OUT0ALT_3/ OPAMP_OUT0ALT ACMP0_CH3	EBI_NANDREn_0/1/2	TIM0_CDTI1_4	US2_RX_0	LES_CH3_0
PC4	DAC0_P0_0/ OPAMP_P0 ACMP0_CH4	EBI_A26_0/1/2	TIM0_CDTI2_4 LETIM0_OUT0_3 PCNT1_S0IN_0	US2_CLK_0 I2C1_SDA_0	LES_CH4_0
PC5	DAC0_N0_0/ OPAMP_N0 ACMP0_CH5	EBI_NANDWEn_0/1/2	LETIM0_OUT1_3 PCNT1_S1IN_0	US2_CS_0 I2C1_SCL_0	LES_CH5_0
PC6	ACMP0_CH6	EBI_A05_0/1/2		LEU1_TX_0 I2C0_SDA_2	LES_CH6_0 ETM_TCLK_2
PC7	ACMP0_CH7	EBI_A06_0/1/2		LEU1_RX_0 I2C0_SCL_2	LES_CH7_0 ETM_TD0_2
PC8	ACMP1_CH0	EBI_A15_0/1/2	TIM2_CC0_2	US0_CS_2	LES_CH8_0
PC9**	ACMP1_CH1	EBI_A09_1/2	TIM2_CC1_2	US0_CLK_2 GPIO_EM4WU2	LES_CH9_0
PC10	ACMP1_CH2	EBI_A10_1/2	TIM2_CC2_2	US0_RX_2	LES_CH10_0
PC11**	ACMP1_CH3	EBI_ALE_1/2		US0_TX_2	LES_CH11_0
PC12**	DAC0_OUT1ALT_0/ OPAMP_OUT1ALT ACMP1_CH4			U1_TX_0	CMU_CLK0_1 LES_CH12_0
PC13	DAC0_OUT1ALT_1/ OPAMP_OUT1ALT ACMP1_CH5		TIM0_CDTI0_1/3 TIM1_CC0_0 TIM1_CC2_4 PCNT0_S0IN_0	U1_RX_0	LES_CH13_0
PC14	DAC0_OUT1ALT_2/ OPAMP_OUT1ALT ACMP1_CH6		TIM0_CDTI1_1/3 TIM1_CC1_0 PCNT0_S1IN_0	US0_CS_3 U0_TX_3	 LES_CH14_0
PC15	DAC0_OUT1ALT_3/ OPAMP_OUT1ALT ACMP1_CH7		TIM0_CDTI2_1/3 TIM1_CC2_0	US0_CLK_3 U0_RX_3	LES_CH15_0 DBG_SWO_1
PD0	ADC0_CH0 DAC0_OUT0ALT_4/ OPAMP_OUT0ALT DAC0_OUT2_1/ OPAMP_OUT2		PCNT2_S0IN_0	US1_TX_1	
PD1	ADC0_CH1 DAC0_OUT1ALT_4/ OPAMP_OUT1ALT		TIM0_CC0_3 PCNT2_S1IN_0	 US1_RX_1	 DBG_SWO_2
PD2	ADC0_CH2	EBI_A27_0/1/2	TIM0_CC1_3	US1_CLK_1	DBG_SWO_3

Continued on next page

Pin Name	Analog	EBI	Timers	Communication USB_DMPU_0	Other
PD3	ADC0_CH3 DAC0_N2_0/ OPAMP_N2		TIM0_CC2_3	US1_CS_1	ETM_TD1_0/2
PD4	ADC0_CH4 DAC0_P2_0/ OPAMP_P2			LEU0_TX_0	ETM_TD2_0/2
PD5	ADC0_CH5 DAC0_OUT2_0/ OPAMP_OUT2			LEU0_RX_0	ETM_TD3_0/2
PD6	ADC0_CH6 DAC0_P1_0/ OPAMP_P1		TIM1_CC0_4 LETIM0_OUT0_0 PCNT0_S0IN_3	US1_RX_2 I2C0_SDA_1	LES_ALTEX0_0 ACMP0_O_2 ETM_TD0_0
PD7	ADC0_CH7 DAC0_N1_0/ OPAMP_N1		TIM1_CC1_4 LETIM0_OUT1_0 PCNT0_S1IN_3	US1_TX_2 I2C0_SCL_1	CMU_CLK0_2 LES_ALTEX1_0 ACMP1_O_2
PD8	BU_VIN				CMU_CLK1_1
PD9**		EBI_CS0_0/1/2			
PD10		EBI_CS1_0/1/2			
PD11**		EBI_CS2_0/1/2			
PD12**		EBI_CS3_0/1/2			
PD13					ETM_TD1_1
PD14				I2C0_SDA_3	
PD15				I2C0_SCL_3	
PE0		EBI_A07_0/1/2	TIM3_CC0_1 PCNT0_S0IN_1	U0_TX_1 I2C1_SDA_2	
PE1		EBI_A08_0/1/2	TIM3_CC1_1 PCNT0_S1IN_1	U0_RX_1 I2C1_SCL_2	
PE2	BU_VOUT	EBI_A09_0	TIM3_CC2_1	U1_TX_3	ACMP0_O_1
PE3	BU_STAT	EBI_A10_0		U1_RX_3	ACMP1_O_1
PE4		EBI_A11_0/1/2		US0_CS_1	
PE5		EBI_A12_0/1/2		US0_CLK_1	
PE6		EBI_A13_0/1/2		US0_RX_1	
PE7		EBI_A14_0/1/2		US0_TX_1	
PE8**		EBI_AD00_0/1/2	PCNT2_S0IN_1		PRS_CH3_1
PE9**		EBI_AD01_0/1/2	PCNT2_S1IN_1		
PE10**		EBI_AD02_0/1/2	TIM1_CC0_1	US0_TX_0	BOOTLOADER_TX
PE11**		EBI_AD03_0/1/2	TIM1_CC1_1	US0_RX_0 BOOTLOADER_RX	LES_ALTEX5_0
PE12**		EBI_AD04_0/1/2	TIM1_CC2_1 US0_CLK_0 I2C0_SDA_6	US0_RX_3 LES_ALTEX6_0	CMU_CLK1_2
PE13**		EBI_AD05_0/1/2	US0_CS_0 I2C0_SCL_6	US0_TX_3 ACMP0_O_0 GPIO_EM4WU5	LES_ALTEX7_0
PE14**		EBI_AD06_0/1/2	TIM3_CC0_0	LEU0_TX_2	
PE15**		EBI_AD07_0/1/2	TIM3_CC1_0	LEU0_RX_2	
PF0			TIM0_CC0_5 LETIM0_OUT0_2	US1_CLK_2 LEU0_TX_3 I2C0_SDA_5	DBG_SWCLK_0/1/2/3
PF1			TIM0_CC1_5 LETIM0_OUT1_2	US1_CS_2 LEU0_RX_3 I2C0_SCL_5	DBG_SWDIO_0/1/2/3 GPIO_EM4WU3

Continued on next page

Pin Name	Analog	EBI	Timers	Communication	Other
PF2		EBI_ARDY_0/1/2	TIM0_CC2_5	LEU0_TX_4	ACMP1_O_0 DBG_SWO_0 GPIO_EM4WU4
PF3		EBI_ALE_0	TIM0_CDTI0_2/5		PRS_CH0_1 ETM_TD3_1
PF4		EBI_WEn_0/2	TIM0_CDTI1_2/5		PRS_CH1_1
PF5		EBI_REn_0/2	TIM0_CDTI2_2/5	USB_VBUSEN_0	PRS_CH2_1
PF6		EBI_BL0_0/1/2	TIM0_CC0_2	U0_TX_0	
PF7		EBI_BL1_0/1/2	TIM0_CC1_2	U0_RX_0	
PF8**		EBI_WEn_1	TIM0_CC2_2		ETM_TCLK_1
PF9**		EBI_REn_1			ETM_TD0_1
PF10				U1_TX_1 USB_DM_0	
PF11				U1_RX_1 USB_DP_0	
PF12				USB_ID_0	
USB_VBUS	USB 5.0 V VBUS input				
USB_VREGI	USB Input to internal 3V3 regulator				
USB_VREGO	USB Decoupling for internal 3V3 USB regulator and regulator output				
VDD_DREG	Power supply for on-chip voltage regulator				
DECOUPLE	Decouple output for on-chip voltage regulator. An external capacitance of size CDECOUPLE is required at this pin				
AVDD_2 VDD_1	Analog power supply 2 Analog power supply 1				
AVDD_0	Analog power supply 0				
RESETn	Reset input Active low, with internal pull-up				

E.2 Alternate functionality pinout

A wide selection of alternate functionality is available for multiplexing to various pins. The table below shows the name of the alternate functionality in the first column, followed by columns showing the possible LOCATION bitfield settings. Some functionality, such as analog interfaces, do not have alternate settings or a LOCATION bitfield. In these cases, the pinout is shown in the column corresponding to LOCATION 0.

Alternate Functionality	LOCATION 0	1	2	3	4	5	6	Description
ACMP0_CH0	PC0							Analog comparator ACMP0, channel 0.
ACMP0_CH1	PC1							Analog comparator ACMP0, channel 1.
ACMP0_CH2	PC2							Analog comparator ACMP0, channel 2.
ACMP0_CH3	PC3							Analog comparator ACMP0, channel 3.
ACMP0_CH4	PC4							Analog comparator ACMP0, channel 4.
ACMP0_CH5	PC5							Analog comparator ACMP0, channel 5.
ACMP0_CH6	PC6							Analog comparator ACMP0, channel 6.
ACMP0_CH7	PC7							Analog comparator ACMP0, channel 7.
ACMP0_O	PE13	PE2	PD6					Analog comparator ACMP0, digital output.
ACMP1_CH0	PC8							Analog comparator ACMP1, channel 0.
ACMP1_CH1	PC9							Analog comparator ACMP1, channel 1.
ACMP1_CH2	PC10							Analog comparator ACMP1, channel 2.
ACMP1_CH3	PC11							Analog comparator ACMP1, channel 3.
ACMP1_CH4	PC12							Analog comparator ACMP1, channel 4.
ACMP1_CH5	PC13							Analog comparator ACMP1, channel 5.
ACMP1_CH6	PC14							Analog comparator ACMP1, channel 6.
ACMP1_CH7	PC15							Analog comparator ACMP1, channel 7.
ACMP1_O	PF2	PE3	PD7					Analog comparator ACMP1, digital output.
ADC0_CH0	PD0							Analog to digital converter ADC0, input channel number 0.
ADC0_CH1	PD1							Analog to digital converter ADC0, input channel number 1.
ADC0_CH2	PD2							Analog to digital converter ADC0, input channel number 2.
ADC0_CH3	PD3							Analog to digital converter ADC0, input channel number 3.
ADC0_CH4	PD4							Analog to digital converter ADC0, input channel number 4.
ADC0_CH5	PD5							Analog to digital converter ADC0, input channel number 5.
ADC0_CH6	PD6							Analog to digital converter ADC0, input channel number 6.
ADC0_CH7	PD7							Analog to digital converter ADC0, input channel number 7.
BOOTLOADER_RX	PE11							Bootloader RX
BOOTLOADER_TX	PE10							Bootloader TX
BU_STAT	PE3							Backup Power Domain status, whether or not system is in backup mode
BU_VIN	PD8							Battery input for Backup Power Domain
BU_VOUT	PE2							Power output for Backup Power Domain
CMU_CLK0	PA2	PC12	PD7					Clock Management Unit, clock output number 0.
CMU_CLK1	PA1	PD8	PE12					Clock Management Unit, clock output number 1.
DAC0_N0 / OPAMP_N0	PC5							Operational Amplifier 0 external negative input.
DAC0_N1 / OPAMP_N1	PD7							Operational Amplifier 1 external negative input.
DAC0_N2 / OPAMP_N2	PD3							Operational Amplifier 2 external negative
DAC0_OUT0 / OPAMP_OUT0	PB11							Digital to Analog Converter DAC0_OUT0 / OPAMP output channel number 0.

Alternate Functionality	LOCATION 0	1	2	3	4	5	6	Description
DAC0_OUT0ALT / OPAMP_OUT0ALT	PC0	PC1	PC2	PC3	PD0			Digital to Analog Converter DAC0_OUT0ALT / OPAMP alternative output for channel 0.
DAC0_OUT1/ OPAMP_OUT1	PB12							Digital to Analog Converter DAC0_OUT1 / OPAMP output channel number 1.
DAC0_OUT1ALT / OPAMP_OUT1ALT	PC12	PC13	PC14	PC15	PD1			Digital to Analog Converter DAC0_OUT1ALT / OPAMP alternative output for channel 1.
DAC0_OUT2 / OPAMP_OUT2	PD5	PD0						Digital to Analog Converter DAC0_OUT2 / OPAMP output channel number 2.
DAC0_P0 / OPAMP_P0	PC4							Operational Amplifier 0 external positive input.
DAC0_P1 / OPAMP_P1	PD6							Operational Amplifier 1 external positive input.
DAC0_P2 / OPAMP_P2	PD4							Operational Amplifier 2 external positive input.
DBG_SWCLK	PF0	PF0	PF0	PF0				Debug-interface Serial Wire clock input. This function is enabled to pin out of reset, and has a built-in pull down.
DBG_SWDIO	PF1	PF1	PF1	PF1				Debug-interface Serial Wire data input / output. This function is enabled to pin out of reset, and has a built-in pull up.
DBG_SWO	PF2	PC15	PD1	PD2				Debug-interface Serial Wire viewer Output. This function is not enabled after reset, and must be enabled by software to be used.
EBI_A00	PA12	PA12	PA12					External Bus Interface (EBI) address output pin 00.
EBI_A01	PA13	PA13	PA13					External Bus Interface (EBI) address output pin 01.
EBI_A02	PA14	PA14	PA14					External Bus Interface (EBI) address output pin 02.
EBI_A03	PB9	PB9	PB9					External Bus Interface (EBI) address output pin 03.
EBI_A04	PB10	PB10	PB10					External Bus Interface (EBI) address output pin 04.
EBI_A05	PC6	PC6	PC6					External Bus Interface (EBI) address output pin 05.
EBI_A06	PC7	PC7	PC7					External Bus Interface (EBI) address output pin 06.
EBI_A07	PE0	PE0	PE0					External Bus Interface (EBI) address output pin 07.
EBI_A08	PE1	PE1	PE1					External Bus Interface (EBI) address output pin 08.
EBI_A09	PE2	PC9	PC9					External Bus Interface (EBI) address output pin 09.
EBI_A10	PE3	PC10	PC10					External Bus Interface (EBI) address output pin 10.
EBI_A11	PE4	PE4	PE4					External Bus Interface (EBI) address output pin 11.
EBI_A12	PE5	PE5	PE5					External Bus Interface (EBI) address output pin 12.
EBI_A13	PE6	PE6	PE6					External Bus Interface (EBI) address output pin 13.
EBI_A14	PE7	PE7	PE7					External Bus Interface (EBI) address output pin 14.
EBI_A15	PC8	PC8	PC8					External Bus Interface (EBI) address output pin 15.
EBI_A16	PB0	PB0	PB0					External Bus Interface (EBI) address output pin 16.
EBI_A17	PB1	PB1	PB1					External Bus Interface (EBI) address output pin 17.
EBI_A18	PB2	PB2	PB2					External Bus Interface (EBI) address output pin 18.
EBI_A19	PB3	PB3	PB3					External Bus Interface (EBI) address output pin 19.
EBI_A20	PB4	PB4	PB4					External Bus Interface (EBI) address output pin 20.
EBI_A21	PB5	PB5	PB5					External Bus Interface (EBI) address output pin 21.
EBI_A22	PB6	PB6	PB6					External Bus Interface (EBI) address output pin 22.
EBI_A23	PC0	PC0	PC0					External Bus Interface (EBI) address output pin 23.
EBI_A24	PC1	PC1	PC1					External Bus Interface (EBI) address output pin 24.
EBI_A25	PC2	PC2	PC2					External Bus Interface (EBI) address output pin 25.
EBI_A26	PC4	PC4	PC4					External Bus Interface (EBI) address output pin 26.
EBI_A27	PD2	PD2	PD2					External Bus Interface (EBI) address output pin 27.
EBI_AD00	PE8	PE8	PE8					External Bus Interface (EBI) address and data input / output pin 00.
EBI_AD01	PE9	PE9	PE9					External Bus Interface (EBI) address and data input / output pin 01.

Alternate Functionality	LOCATION 0	1	2	3	4	5	6	Description
EBI_AD02	PE10	PE10	PE10					External Bus Interface (EBI) address and data input / output pin 02.
EBI_AD03	PE11	PE11	PE11					External Bus Interface (EBI) address and data input / output pin 03.
EBI_AD04	PE12	PE12	PE12					External Bus Interface (EBI) address and data input / output pin 04.
EBI_AD05	PE13	PE13	PE13					External Bus Interface (EBI) address and data input / output pin 05.
EBI_AD06	PE14	PE14	PE14					External Bus Interface (EBI) address and data input / output pin 06.
EBI_AD07	PE15	PE15	PE15					External Bus Interface (EBI) address and data input / output pin 07.
EBI_AD08	PA15	PA15	PA15					External Bus Interface (EBI) address and data input / output pin 08.
EBI_AD09	PA0	PA0	PA0					External Bus Interface (EBI) address and data input / output pin 09.
EBI_AD10	PA1	PA1	PA1					External Bus Interface (EBI) address and data input / output pin 10.
EBI_AD11	PA2	PA2	PA2					External Bus Interface (EBI) address and data input / output pin 11.
EBI_AD12	PA3	PA3	PA3					External Bus Interface (EBI) address and data input / output pin 12.
EBI_AD13	PA4	PA4	PA4					External Bus Interface (EBI) address and data input / output pin 13.
EBI_AD14	PA5	PA5	PA5					External Bus Interface (EBI) address and data input / output pin 14.
EBI_AD15	PA6	PA6	PA6					External Bus Interface (EBI) address and data input / output pin 15.
EBI_ALE	PF3	PC11	PC11					External Bus Interface (EBI) Address Latch Enable output.
EBI_ARDY	PF2	PF2	PF2					External Bus Interface (EBI) Hardware Ready Control input.
EBI_BL0	PF6	PF6	PF6					External Bus Interface (EBI) Byte Lane/Enable pin 0.
EBI_BL1	PF7	PF7	PF7					External Bus Interface (EBI) Byte Lane/Enable pin 1.
EBI_CS0	PD9	PD9	PD9					External Bus Interface (EBI) Chip Select output 0.
EBI_CS1	PD10	PD10	PD10					External Bus Interface (EBI) Chip Select output 1.
EBI_CS2	PD11	PD11	PD11					External Bus Interface (EBI) Chip Select output 2.
EBI_CS3	PD12	PD12	PD12					External Bus Interface (EBI) Chip Select output 3.
EBI_NANDREn	PC3	PC3	PC3					External Bus Interface (EBI) NAND Read Enable output.
EBI_NANDWEn	PC5	PC5	PC5					External Bus Interface (EBI) NAND Write Enable output.
EBI_REn	PF5	PF9	PF5					External Bus Interface (EBI) Read Enable output.
EBI_WEn	PF4	PF8	PF4					External Bus Interface (EBI) Write Enable output.
ETM_TCLK	PD7	PF8	PC6	PA6				Embedded Trace Module ETM clock .
ETM_TD0	PD6	PF9	PC7	PA2				Embedded Trace Module ETM data 0.
ETM_TD1	PD3	PD13	PD3	PA3				Embedded Trace Module ETM data 1.
ETM_TD2	PD4	PB15	PD4	PA4				Embedded Trace Module ETM data 2.
ETM_TD3	PD5	PF3	PD5	PA5				Embedded Trace Module ETM data 3.
GPIO_EM4WU0	PA0							Pin can be used to wake the system up from EM4
GPIO_EM4WU1	PA6							Pin can be used to wake the system up from EM4
GPIO_EM4WU2	PC9							Pin can be used to wake the system up from EM4
GPIO_EM4WU3	PF1							Pin can be used to wake the system up from EM4
GPIO_EM4WU4	PF2							Pin can be used to wake the system up from EM4
GPIO_EM4WU5	PE13							Pin can be used to wake the system up from EM4
HFXTAL_N	PB14							High Frequency Crystal (4 - 48 MHz) negative pin. Also used as external optional clock input pin.
HFXTAL_P	PB13							High Frequency Crystal (4 - 48 MHz) positive pin.

Alternate Functionality	LOCATION 0	1	2	3	4	5	6	Description
I2C0_SCL	PA1	PD7	PC7	PD15	PC1	PF1	PE13	I2C0 Serial Clock Line input / output.
I2C0_SDA	PA0	PD6	PC6	PD14	PC0	PF0	PE12	I2C0 Serial Data input / output.
I2C1_SCL	PC5	PB12	PE1					I2C1 Serial Clock Line input / output.
I2C1_SDA	PC4	PB11	PE0					I2C1 Serial Data input / output.
LES_ALTEX0	PD6							LESENSE alternate exite output 0.
LES_ALTEX1	PD7							LESENSE alternate exite output 1.
LES_ALTEX2	PA3							LESENSE alternate exite output 2.
LES_ALTEX3	PA4							LESENSE alternate exite output 3.
LES_ALTEX4	PA5							LESENSE alternate exite output 4.
LES_ALTEX5	PE11							LESENSE alternate exite output 5.
LES_ALTEX6	PE12							LESENSE alternate exite output 6.
LES_ALTEX7	PE13							LESENSE alternate exite output 7.
LES_CH0	PC0							LESENSE channel 0.
LES_CH1	PC1							LESENSE channel 1.
LES_CH2	PC2							LESENSE channel 2.
LES_CH3	PC3							LESENSE channel 3.
LES_CH4	PC4							LESENSE channel 4.
LES_CH5	PC5							LESENSE channel 5.
LES_CH6	PC6							LESENSE channel 6.
LES_CH7	PC7							LESENSE channel 7.
LES_CH8	PC8							LESENSE channel 8.
LES_CH9	PC9							LESENSE channel 9.
LES_CH10	PC10							LESENSE channel 10.
LES_CH11	PC11							LESENSE channel 11.
LES_CH12	PC12							LESENSE channel 12.
LES_CH13	PC13							LESENSE channel 13.
LES_CH14	PC14							LESENSE channel 14.
LES_CH15	PC15							LESENSE channel 15.
LETIM0_OUT0	PD6	PB11	PF0	PC4				Low Energy Timer LETIM0, output channel 0.
LETIM0_OUT1	PD7	PB12	PF1	PC5				Low Energy Timer LETIM0, output channel 1.
LEU0_RX	PD5	PB14	PE15	PF1	PA0			LEUART0 Receive input.
LEU0_TX	PD4	PB13	PE14	PF0	PF2			LEUART0 Transmit output. Also used as receive input in half duplex communication.
LEU1_RX	PC7	PA6						LEUART1 Receive input.
LEU1_TX	PC6	PA5						LEUART1 Transmit output. Also used as receive input in half duplex communication.

Alternate Functionality	LOCATION 0	1	2	3	4	5	6	Description
LFXTAL_N	PB8							Low Frequency Crystal (typically 32.768 kHz) negative pin. Also used as an optional external clock input pin.
LFXTAL_P	PB7							Low Frequency Crystal (typically 32.768 kHz) positive pin.
PCNT0_S0IN	PC13	PE0	PC0	PD6				Pulse Counter PCNT0 input number 0.
PCNT0_S1IN	PC14	PE1	PC1	PD7				Pulse Counter PCNT0 input number 1.
PCNT1_S0IN	PC4	PB3						Pulse Counter PCNT1 input number 0.
PCNT1_S1IN	PC5	PB4						Pulse Counter PCNT1 input number 1.
PCNT2_S0IN	PD0	PE8						Pulse Counter PCNT2 input number 0.
PCNT2_S1IN	PD1	PE9						Pulse Counter PCNT2 input number 1.
PRS_CH0	PA0	PF3						Peripheral Reflex System PRS, channel 0.
PRS_CH1	PA1	PF4						Peripheral Reflex System PRS, channel 1.
PRS_CH2	PC0	PF5						Peripheral Reflex System PRS, channel 2.
PRS_CH3	PC1	PE8						Peripheral Reflex System PRS, channel 3.
TIM0_CC0	PA0	PA0	PF6	PD1	PA0	PF0		Timer 0 Capture Compare input / output channel 0.
TIM0_CC1	PA1	PA1	PF7	PD2	PC0	PF1		Timer 0 Capture Compare input / output channel 1.
TIM0_CC2	PA2	PA2	PF8	PD3	PC1	PF2		Timer 0 Capture Compare input / output channel 2.
TIM0_CDTI0	PA3	PC13	PF3	PC13	PC2	PF3		Timer 0 Complimentary Deat Time Insertion channel 0.
TIM0_CDTI1	PA4	PC14	PF4	PC14	PC3	PF4		Timer 0 Complimentary Deat Time Insertion channel 1.
TIM0_CDTI2	PA5	PC15	PF5	PC15	PC4	PF5		Timer 0 Complimentary Deat Time Insertion channel 2.
TIM1_CC0	PC13	PE10	PB0	PB7	PD6			Timer 1 Capture Compare input / output channel 0.
TIM1_CC1	PC14	PE11	PB1	PB8	PD7			Timer 1 Capture Compare input / output channel 1.
TIM1_CC2	PC15	PE12	PB2	PB11	PC13			Timer 1 Capture Compare input / output channel 2.
TIM2_CC0	PA8	PA12	PC8					Timer 2 Capture Compare input / output channel 0.
TIM2_CC1	PA9	PA13	PC9					Timer 2 Capture Compare input / output channel 1.
TIM2_CC2	PA10	PA14	PC10					Timer 2 Capture Compare input / output channel 2.
TIM3_CC0	PE14	PE0						Timer 3 Capture Compare input / output channel 0.
TIM3_CC1	PE15	PE1						Timer 3 Capture Compare input / output channel 1.
TIM3_CC2	PA15	PE2						Timer 3 Capture Compare input / output channel 2.
U0_RX	PF7	PE1	PA4	PC15				UART0 Receive input.
U0_TX	PF6	PE0	PA3	PC14				UART0 Transmit output. Also used as receive input in half duplex communication.
U1_RX	PC13	PF11	PB10	PE3				UART1 Receive input.
U1_TX	PC12	PF10	PB9	PE2				UART1 Transmit output. Also used as receive input in half duplex communication.
US0_CLK	PE12	PE5	PC9	PC15	PB13	PB13		USART0 clock input / output.
US0_CS	PE13	PE4	PC8	PC14	PB14	PB14		USART0 chip select input / output.
US0_RX	PE11	PE6	PC10	PE12	PB8	PC1		USART0 Asynchronous Receive. USART0 Synchronous mode Master Input / Slave Output (MISO).
US0_TX	PE10	PE7	PC11	PE13	PB7	PC0		USART0 Asynchronous Transmit.Also used as receive input in half duplex communication. USART0 Synchronous mode Master Output / Slave Input (MOSI).
US1_CLK	PB7	PD2	PF0					USART1 clock input / output.
US1_CS	PB8	PD3	PF1					USART1 chip select input / output.
US1_RX	PC1	PD1	PD6					USART1 Asynchronous Receive. USART1 Synchronous mode Master Input / Slave Output (MISO).
US1_TX	PC0	PD0	PD7					USART1 Asynchronous Transmit.Also used as receive input in half duplex communication. USART1 Synchronous mode Master Output / Slave Input (MOSI)

Alternate Functionality	LOCATION 0	1	2	3	4	5	6	Description
US2_CLK	PC4	PB5						USART2 clock input / output.
US2_CS	PC5	PB6						USART2 chip select input / output.
US2_RX	PC3	PB4						USART2 Asynchronous Receive. USART2 Synchronous mode Master Input / Slave Output (MISO).
US2_TX	PC2	PB3						USART2 Asynchronous Transmit.Also used as receive input in half duplex communication. USART2 Synchronous mode Master Output / Slave Input (MOSI).
USB_DM	PF10							USB D- pin.
USB_DMPU	PD2							USB D- Pullup control.
USB_DP	PF11							USB D+ pin.
USB_ID	PF12							USB ID pin. Used in OTG mode.
USB_VBUS	USB_VBUS							USB 5V VBUS input.
USB_VBUSEN	PF5							USB 5V VBUS enable.
USB_VREGI	USB_VREGI							USB Input to internal 3v3 regulator
USB_VREGO	USV_VREGO							USB Decoupling for internal 3v3 USB regulator and regulator output.

F ARM memory and bus system

The ARM core contains an AMBA AHB Bus system allowing bus masters to access the memory mapped address space. A multilayer AHB bus matrix, using a Round-robin arbitration scheme, connects the master bus interfaces to the AHB slaves (Figure 38). 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 AHB bus masters are:

- **Cortex-M3 ICode:** Used for instruction fetches from Code memory (0x00000000 - 0x1FFFFFFF).
- **Cortex-M3 DCode:** Used for debug and data access to Code memory (0x00000000 - 0x1FFFFFFF).
- **Cortex-M3 System:** Used for instruction fetches, data and debug access to system space (0x20000000 - 0xDFFFFFFF).
- **DMA:** Can access SRAM, Flash and peripherals (0x00000000 - 0xDFFFFFFF).
- **USB DMA:** Can access SRAM and Flash (0x80000000 - 0xDFFFFFFF, 0x00000000 - 0x3FFFFFFF), and the AHB-peripherals: USB and AES.

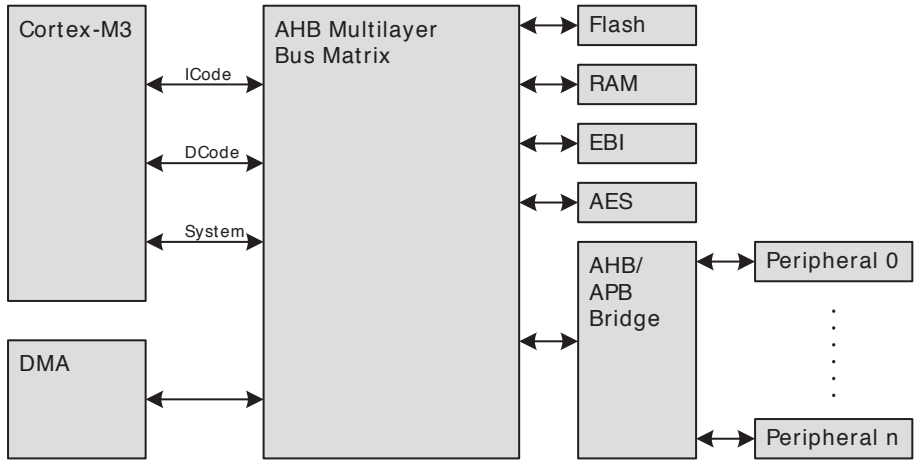


Figure 38: ARM core bus system

F.1 Functional description

The memory segments are mapped together with the internal segments of the Cortex-M3 into the system memory map shown by Figure 39

The embedded SRAM is located at address 0x20000000 in the memory map of the ARM core. When running code located in SRAM starting at this address, the Cortex-M3 uses the System bus to fetch instructions. This results in reduced performance as the Cortex-M3 accesses stack, other data in SRAM and peripherals

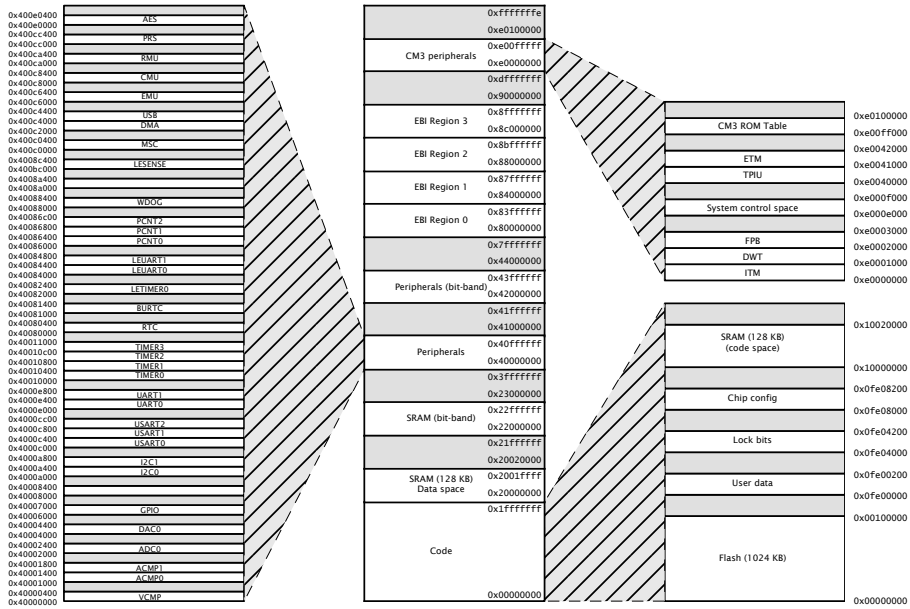


Figure 39: Memory Map

using the System bus. 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-M3 fetches instructions through the I/D-Code bus interface, leaving the System bus for data access. The SRAM mapped into the code space can however only be accessed by the CPU, i.e. not the DMA.

F.1.1 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 ARM core.

The standard approach to modify a single register or SRAM bit in the aliased regions, requires 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 read-modify-write can be done in a single atomic operation. 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 allows addressing each individual bit in the SRAM and peripheral areas of the memory map. To set or clear a bit in the embedded SRAM, write a 1 or a 0 to the following address:

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

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) × 32 + *bit* × 4

where *address* and *bit* are defined as above.

Note that the AHB-peripherals USB and AES does not support bit-banding.

F.2 Peripherals

The peripherals are mapped into the peripheral memory segment, each with a fixed size address range according to Table 4, Table 5 and Table 6.

Table 4: Core peripherals

Address range	Peripheral
0x400E0400 - 0x41FFFFFF	Reserved
0x400E0000 - 0x400E03FF	AES
0x400CC400 - 0x400FFFFF	Reserved
0x400CC000 - 0x400CC3FF	PRS
0x400CA400 - 0x400CBFFF	Reserved
0x400CA000 - 0x400CA3FF	RMU
0x400C8400 - 0x400C9FFF	Reserved
0x400C8000 - 0x400C83FF	CMU
0x400C6400 - 0x400C7FFF	Reserved
0x400C6000 - 0x400C63FF	EMU
0x400C4400 - 0x400C5FFF	Reserved
0x400C4000 - 0x400C43FF	USB
0x400C2000 - 0x400C3FFF	DMA
0x400C0400 - 0x400C1FFF	Reserved
0x400C0000 - 0x400C03FF	MSC

Address range	Peripheral
0x4008C400 - 0x400BFFFF	Reserved
0x4008C000 - 0x4008C3FF	LESENSE
0x4008A400 - 0x4008BFFF	Reserved
0x4008A000 - 0x4008A3FF	Reserved
0x40088400 - 0x40089FFF	Reserved
0x40088000 - 0x400883FF	WDOG
0x40086C00 - 0x40087FFF	Reserved
0x40086800 - 0x40086BFF	PCNT2
0x40086400 - 0x400867FF	PCNT1
0x40086000 - 0x400863FF	PCNT0
0x40084800 - 0x40085FFF	Reserved
0x40084400 - 0x400847FF	LEUART1
0x40084000 - 0x400843FF	LEUART0
0x40082400 - 0x40083FFF	Reserved
0x40082000 - 0x400823FF	LETIMER0
0x40081400 - 0x40081FFF	Reserved
0x40081000 - 0x400813FF	BCKRTC
0x40080400 - 0x40080FFF	Reserved
0x40080000 - 0x400803FF	RTC

Table 5: Low energy peripherals

Address range	Peripheral
0x40011000 - 0x4007FFFF	Reserved
0x40010C00 - 0x40010FFF	TIMER3
0x40010800 - 0x40010BFF	TIMER2
0x40010400 - 0x400107FF	TIMER1
0x40010000 - 0x400103FF	TIMER0
0x4000E800 - 0x4000FFFF	Reserved
0x4000E400 - 0x4000E7FF	UART1
0x4000E000 - 0x4000E3FF	UART0
0x4000CC00 - 0x4000DFFF	Reserved
0x4000C800 - 0x4000CBFF	USART2
0x4000C400 - 0x4000C7FF	USART1
0x4000C000 - 0x4000C3FF	USART0
0x4000A800 - 0x4000BFFF	Reserved
0x4000A400 - 0x4000A7FF	I2C1
0x4000A000 - 0x4000A3FF	I2C0
0x40008400 - 0x40009FFF	Reserved
0x40008000 - 0x400083FF	EBI
0x40007000 - 0x40007FFF	Reserved
0x40006000 - 0x40006FFF	GPIO
0x40004400 - 0x40005FFF	Reserved
0x40004000 - 0x400043FF	DAC0
0x40002400 - 0x40003FFF	Reserved
0x40002000 - 0x400023FF	ADC0
0x40001800 - 0x40001FFF	Reserved
0x40001400 - 0x400017FF	ACMP1
0x40001000 - 0x400013FF	ACMP0
0x40000400 - 0x40000FFF	Reserved
0x40000000 - 0x400003FF	VCMP

Table 6: Low energy peripherals

F.2.1 Interrupt operation

The ARM-core has up to 31 interrupt request lines (IRQ) which are connected to the Cortex-M3. Each of these lines 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-M3 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.

Interrupt Request Lines (IRQ)	Source
0	DMA
1	GPIO_EVEN
2	TIMER0
3	USART0_RX
4	USART0_TX
5	USB
6	ACMP0/ACMP1
7	ADC0
8	DAC0
9	I2C0
10	I2C1
11	GPIO_ODD
12	TIMER1
13	TIMER2
14	TIMER3
15	USART1_RX
16	USART1_TX
17	LESENSE
18	USART2_RX
19	USART2_TX
20	UART0_RX
21	UART0_TX
22	UART1_RX
23	UART1_TX
24	LEUART0
25	LEUART1
26	LETIMER0
27	PCNT0
28	PCNT1
29	PCNT2
30	RTC
31	BURTC
32	CMU
33	VCMP
34	Reserved
35	MSC
36	AES
37	EBI

Table 7: Interrupts

F.3 Bus Matrix

The Bus Matrix connects the memory segments to the bus masters:

- Code: CPU instruction or data fetches from the code space

- System: CPU read and write to the SRAM, EBI and peripherals
- DMA: Access to SRAM, EBI, Flash and peripherals
- USB DMA: Access to SRAM, EBI and Flash

F.3.1 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.

F.3.2 Access Performance

The Bus Matrix is a multi-layer energy optimized AMBA AHB compliant bus with an internal bandwidth equal to 4 times a single AHB-bus.

The Bus Matrix accepts new transfers initiated by each master in every clock cycle without inserting any wait-states. The slaves, however, may insert wait-states depending on their internal throughput and the clock frequency.

The Cortex-M3, the DMA Controller, and the peripherals run on clocks that can be prescaled separately. When accessing a peripheral which runs on a frequency equal to or faster than the HFCORECLK, the number of wait cycles per access, in addition to master arbitration, is given by:

$N_{cycles} = 2 + N_{slave\ cycles}$

where $N_{slave\ cycles}$ is the wait cycles introduced by the slave.

When accessing a peripheral running on a clock slower than the HFCORECLK, wait-cycles are introduced to allow the transfer to complete on the peripheral clock. The number of wait cycles per access, in addition to master arbitration, is given by:

$N_{cycles} = (2 + N_{slave\ cycles}) \times f_{HFCORECLK}/f_{HFPERCLK}$,

where $N_{slave\ cycles}$ is the number of wait cycles introduced by the slave.

For general register access:

$N_{slave\ cycles} = 1$.

F.4 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 and in some cases also EM3. This enables the peripherals to perform tasks while the system energy consumption is minimal.

The Low Energy Peripherals are:

- Low Energy Timer - LETIMER
- Low Energy UART - LEUART
- Pulse Counter - PCNT
- Real Time Counter - RTC
- Watchdog - WDOG
- Low Energy Sensor Interface - LESENSE
- Backup RTC - BURTC

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

F.4.1 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 ARM core; immediate synchronization, and delayed synchronization. Immediate synchronization is available for the RTC, LETIMER and LESENSE, and results in an immediate update of the target registers. Delayed synchronization is used for the other Low Energy Peripherals, and for these peripherals, a write operation requires 3 positive edges on the clock of the Low Energy Peripheral being accessed. Registers requiring synchronization are marked "Asynchronous" in their description header.

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. LEUART_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 can be entered immediately after writing a register.

See Figure 40 for a more detailed overview of the write operation.

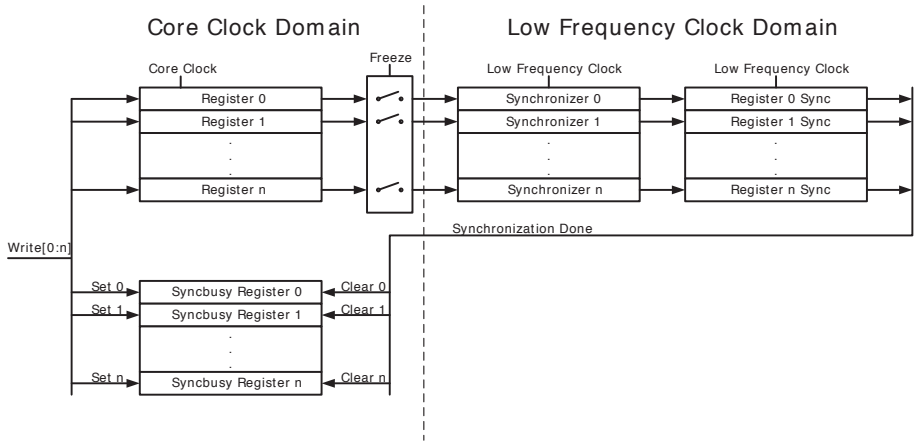


Figure 40: Write operation to Low Energy Peripherals

Immediate synchronization

Contrary to the peripherals with delayed synchronization, data written to peripherals with immediate synchronization, takes effect in the peripheral immediately. They are updated immediately on the peripheral write access. If a write is set up close to a peripheral clock edge, the write is delayed to after the clock edge. This will introduce wait-states on peripheral access. In the worst case, there can be three wait-state cycles of the HFCORECLK_LE and an additional wait-state equivalent of up to 315 ns.

For peripherals with immediate synchronization, the SYNCBUSY registers are still present. Commands written to a peripheral with immediate synchronization are not executed before the first peripheral clock after the write. During this period, the SYNCBUSY flag in the command register is set, indicating that the command has not yet been executed.

F.4.2 Reading

When reading from Low Energy Peripherals, the data is synchronized regardless of the originating clock domain. Registers updated/maintained by the Low Energy Peripheral are read directly from the Low Energy clock domain. Registers residing in the core clock domain, are read from the core clock domain. See Figure 41 for a more detailed overview of the read operation.

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

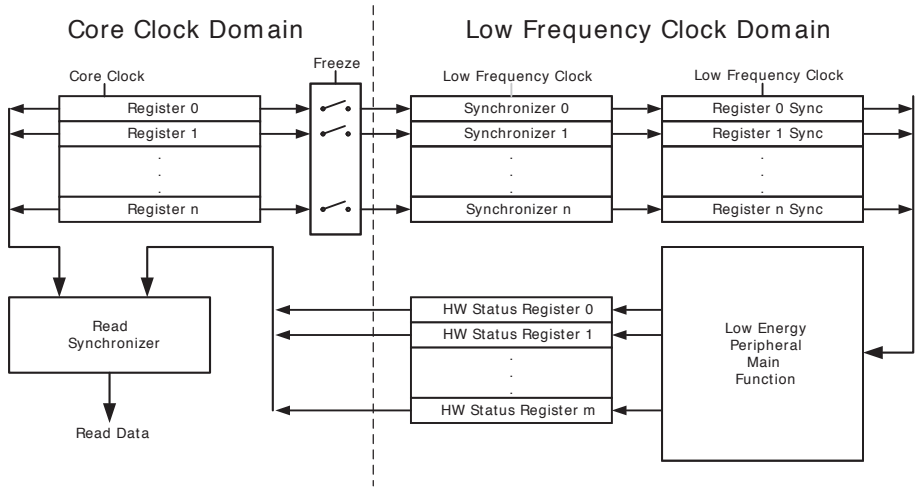


Figure 41: Read operation from Low Energy Peripherals

F.5 FREEZE register

For Low Energy Peripherals with delayed synchronization there is a <module_name>_FREEZE register (e.g. RTC_FREEZE), containing 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 has no effect.

F.6 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 4096 bytes (minimum erase unit)
- Minimum 20,000 erase cycles
- More than 10 years data retention at 85°C
- Lock-bits for memory protection
- Data retention in any state

F.7 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 used to transfer data between the SRAM, Flash and peripherals.

- Up to 128 kB memory
- Bit-band access support
- 32 kB blocks may be individually powered down when not in use
- Data retention of the entire memory in EM0 to EM3

F.8 Device Information (DI) Page

The DI page contains calibration values, a unique identification number and other useful data. See the table below for a complete overview.

DI Address	Register	Description
0x0FE08020	CMU_LFRCOCTRL	Register reset value
0x0FE08028	CMU_HFRCOCTRL	Register reset value
0x0FE08030	CMU_AUXHFRCOCTRL	Register reset value
0x0FE08040	ADC0_CAL	Register reset value
0x0FE08048	ADC0_BIASPROG	Register reset value
0x0FE08050	DAC0_CAL	Register reset value
0x0FE08058	DAC0_BIASPROG	Register reset value
0x0FE08060	ACMP0_CTRL	Register reset value
0x0FE08068	ACMP1_CTRL	Register reset value
0x0FE08078	CMU_LCDCTRL	Register reset value
0x0FE080A0	DAC0_OPACTRL	Register reset value
0x0FE080A8	DAC0_OPAOFFSET	Register reset value
0x0FE081B0	DI_CRC	[15:0]: DI data CRC-16
0x0FE081B2	CAL_TEMP_0	[7:0] Calibration temperature (DegC)
0x0FE081B3	RESERVED	[7:0]: Reserved for other temperature information
0x0FE081B4	ADC0_CAL_1V25	[14:8]: Gain for 1V25 reference [6:0]: Offset for 1V25 reference
0x0FE081B6	ADC0_CAL_2V5	[14:8]: Gain for 2V5 reference [6:0]: Offset for 2V5 reference
0x0FE081B8	ADC0_CAL_VDD	[14:8]: Gain for VDD reference [6:0]: Offset for VDD reference
0x0FE081BA	ADC0_CAL_5VDIFF	[14:8]: Gain for 5VDIFF reference [6:0]: Offset for 5VDIFF reference
0x0FE081BC	ADC0_CAL_2XVDD	[14:8]: Reserved (gain for this reference cannot be calibrated) [6:0]: Offset for 2XVDD reference
0x0FE081BE	ADC0_TEMP_0_READ_1V25	[15:4] Temperature reading at 1V25 reference [3:0] Reserved
0x0FE081C0	RESERVED	[15:0] Reserved for other temperature information
0x0FE081C2	RESERVED	[15:0] Reserved
0x0FE081C4	RESERVED	[31:0] Reserved
0x0FE081C8	DAC0_CAL_1V25	[22:16]: Gain for 1V25 reference [13:8]: Channel 1 offset for 1V25 reference [5:0]: Channel 0 offset for 1V25 reference
0x0FE081CC	DAC0_CAL_2V5	[22:16]: Gain for 2V5 reference [13:8]: Channel 1 offset for 2V5 reference [5:0]: Channel 0 offset for 2V5 reference
0x0FE081D0	DAC0_CAL_VDD	[22:16]: Reserved (gain for this reference cannot be calibrated) [13:8]: Channel 1 offset for VDD reference [5:0]: Channel 0 offset for VDD reference
0x0FE081D4	AUXHFRCO_CALIB_BAND_1	[7:0]: 1 MHz tuning
0x0FE081D5	AUXHFRCO_CALIB_BAND_7	[7:0]: 7 MHz tuning
0x0FE081D6	AUXHFRCO_CALIB_BAND_11	[7:0]: 11 MHz tuning
0x0FE081D7	AUXHFRCO_CALIB_BAND_14	[7:0]: 14 MHz tuning
0x0FE081D8	AUXHFRCO_CALIB_BAND_21	[7:0]: 21 MHz tuning
0x0FE081D9	AUXHFRCO_CALIB_BAND_28	[7:0]: 28 MHz tuning
0x0FE081DC	HFRCO_CALIB_BAND_1	[7:0]: 1 MHz tuning
0x0FE081DD	HFRCO_CALIB_BAND_7	[7:0]: 7 MHz tuning
0x0FE081DE	HFRCO_CALIB_BAND_11	[7:0]: 11 MHz tuning
0x0FE081DF	HFRCO_CALIB_BAND_14	[7:0]: 14 MHz tuning
0x0FE081E0	HFRCO_CALIB_BAND_21	[7:0]: 21 MHz tuning
0x0FE081E1	HFRCO_CALIB_BAND_28	[7:0]: 28 MHz tuning
0x0FE081E2	RESERVED	[15:0] Reserved
0x0FE081E4	RESERVED	23:0] Reserved
0x0FE081E7	MEM_INFO_PAGE_SIZE	[7:0] Flash page size in bytes coded as $2^{(MEM\_INFO\_PAGE\_SIZE + 10)\&0xFF)}$
0x0FE081F0	UNIQUE_0	[31:0] Unique number
0x0FE081F4	UNIQUE_1	[63:32] Unique number
0x0FE081F8	MEM_INFO_FLASH	[15:0]: Flash size, kbyte count as unsigned integer (e.g. 128)
0x0FE081FA	MEM_INFO_RAM	[15:0]: Ram size, kbyte count as unsigned integer (e.g. 16)

Table 8: Device Information Page

G ARM debug interface

The ARM core includes hardware debug support through a 2-pin serial-wire debug (SWD) interface and an Embedded Trace Module (ETM) for data/instruction tracing. In addition, there is also a Serial Wire Viewer pin which can be used to output profiling information, data trace and software-generated messages.

G.1 Features

- 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

G.2 Functional description

There are three debug pins and four trace pins available on the device. Operation of these pins are described in the following section.

G.2.1 Debug pins

The following pins are the debug connections for the device:

- Serial Wire Clock input (SWCLK): This pin is enabled after reset and has a built-in pull down.
- Serial Wire Data Input/Output (SWDIO): This pin is enabled after reset and has a built-in pull-up.
- Serial Wire Viewer (SWV): This pin is disabled after reset.

The debug pins can be enabled and disabled through GPIO_ROUTE. Please remember that upon disabling, debug contact with the device is lost. Also note that, because the debug pins have pull-down and pull-up enabled by default, leaving them enabled might increase the current consumption with up to 200 μA if left connected to supply or ground.

G.2.2 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. Before enabling the ETM, the AUXHFRCO clock needs to be enabled by setting AUXHFRCOEN in CMU_OSCENCMD. 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.

G.2.3 Debug and EM2/EM3

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

G.3 Debug Lock and Device Erase

The debug access to the Cortex-M3 is locked by clearing the Debug Lock Word (DLW) and resetting the device.

When debug access is locked, the debug interface remains accessible but the connection to the Cortex-M3 core and the whole bus-system is blocked as shown in Figure 43. This mechanism is controlled by the Authentication Access Port (AAP) as illustrated by Figure 42. The AAP is only accessible from a debugger and not from the core.

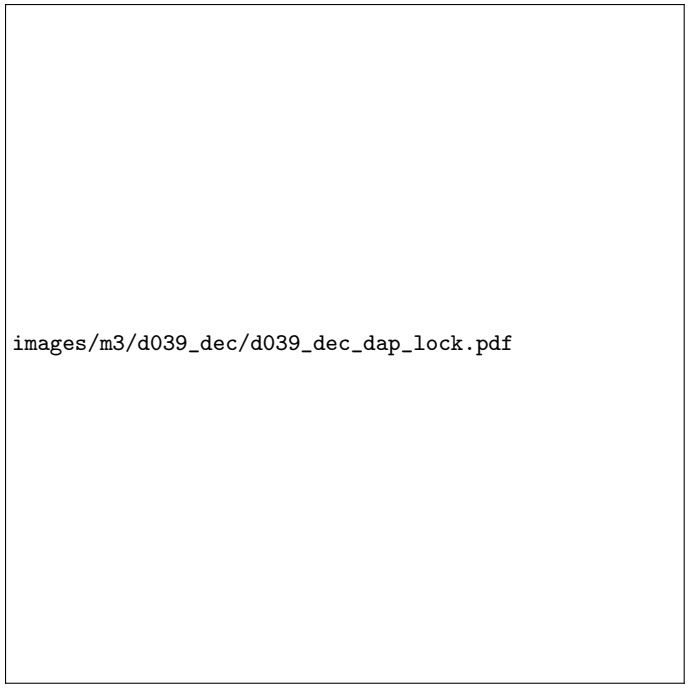


Figure 42: AAP - Authentication Access Port

The debugger can access the AAP-registers, and only these registers just after reset, for the time of the AAP-window outlined in Figure 43. If the device is locked, access to the core and bus-system is blocked even after code execution starts, and the debugger can only access the AAP-registers. If the device is not locked, the AAP is no longer accessible after code execution starts, and the debugger can access the core and bus-system normally. The AAP window can be extended by issuing the bit pattern on SWDIO/SWCLK as shown in Figure 44. This pattern should be applied just before reset is deasserted, and will give the debugger more time to access the AAP.

Figure 43: Device Unlock

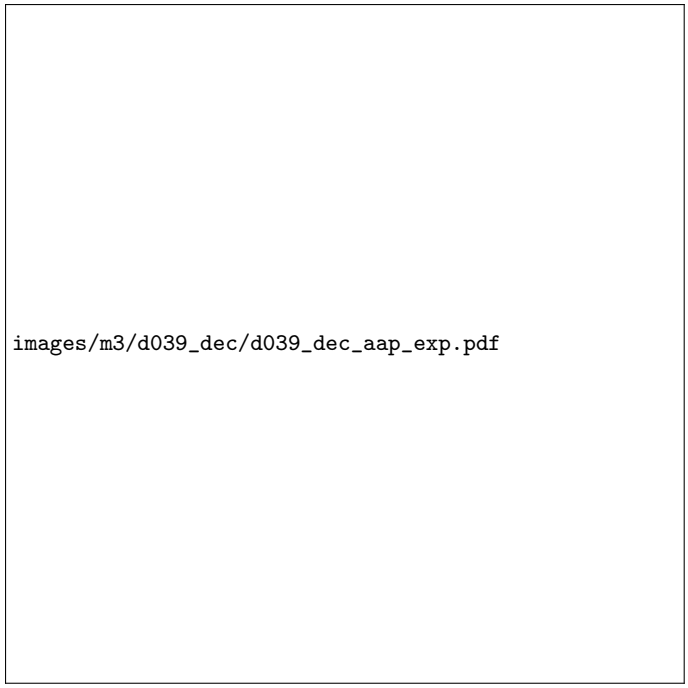


Figure 44: AAP Expansion

If the device is locked, 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. The commands are not executed before AAP_CMDKEY is invalidated, so this register should be cleared to to start the erase operation. This operation erases the main block of flash, all lock bits are reset and debug access through the AHB-AP is enabled. The operation takes 125 ms to complete. Note that the SRAM contents will also be deleted during a device erase, while the UD-page is not erased.

Even if the device is not locked, the can device can be erased through the AAP, using the above procedure during the AAP window. This can be useful if the device has been programmed with code that, e.g., disables the debug interface pins on start-up, or does something else that prevents communication with a debugger.

If the device is locked, the debugger may read the status 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. If the device is not locked, the device erase starts when the AAP window closes, so it is not possible to poll the status.

G.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
0x0FC	AAP_IDR	R	AAP Identification Register

G.5 Register Description

G.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	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.			

G.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. After AAP_CMD is written, this register should be cleared to execute the command.			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0xCFACC118	WRITEEN	Enable write to AAP_CMD

G.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
Access																																R
Name																																ERASEBUSY

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	ERASEBUSY	0	R	**Device Erase Command Status**
	This bit is set when a device erase is executing.			

G.5.4 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	0x16E60001																															
Access	R																															
Name	ID																															

Bit	Name	Reset	Access	Description
31:0	ID	0x16E60001	R	**AAP Identification Register**
	Access port identification register in compliance with the ARM ADI v5 specification (JEDEC Manufacturer ID) .			

H ARM Direct Memory Access Controller

H.1 Introduction

The Direct Memory Access (DMA) controller performs memory 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 for example when moving data from the USART to RAM. The DMA controller uses the PL230 μDMA controller licensed from ARM[1]. Each of the PL230s channels on the ARM core can be connected to any of the ARM peripherals.

H.2 Features

- The DMA controller is accessible as a memory mapped peripheral
- Possible data transfers include
  - RAM/Flash to peripheral
  - RAM to Flash
  - Peripheral to RAM
  - RAM/Flash to RAM
- The DMA controller has 12 independent channels
- Each channel has one (primary) or two (primary and alternate) descriptors
- The configuration for each channel includes
  - Transfer mode
  - Priority
  - Word-count
  - Word-size (8, 16, 32 bit)
- The transfer modes include
  - Basic (using the primary or alternate DMA descriptor)
  - Ping-pong (switching between the primary or alternate DMA descriptors, for continuous data flow to/from peripherals)
  - Scatter-gather (using the primary descriptor to configure the alternate descriptor)
- Each channel has a programmable transfer length
- Channels 0 and 1 support looped transfers
- Channel 0 supports 2D copy
- A DMA channel can be triggered by any of several sources:

[1] ARM PL230 homepage

- Communication modules (USART, UART, LEUART)
- Timers (TIMER)
- Analog modules (DAC, ACMP, ADC)
- Software

- Programmable mapping between channel number and peripherals - any DMA channel can be triggered by any of the available sources
- Interrupts upon transfer completion
- Data transfer to/from LEUART in EM2 is supported by the DMA, providing extremely low energy consumption while performing UART communications

H.3 Block Diagram

An overview of the DMA and the modules it interacts with is shown in Figure 45.

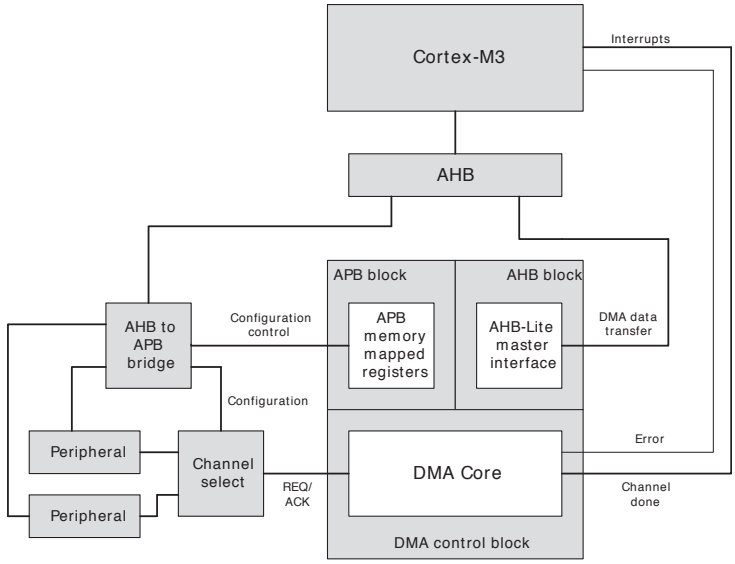


Figure 45: DMA Block Diagram

The DMA Controller consists of four main parts:

- An APB block allowing software to configure the DMA controller
- An AHB block allowing the DMA to read and write the DMA descriptors and the source and destination data for the DMA transfers
- A DMA control block controlling the operation of the DMA, including request/acknowledge signals for the connected peripherals
- A channel select block routing the right peripheral request to each DMA channel

H.4 Functional Description

The 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 DMA work autonomously with the LEUART for data transfer in EM2 without having to wake up the processor core from sleep.

The DMA Controller contains 12 independent channels. Each of these channels can be connected to any of the available peripheral trigger sources by writing to the configuration registers, see Section H.4.1. In addition, each channel can be triggered by software (for large memory transfers or for debugging purposes).

What the DMA Controller should do (when one of its channels is triggered) is configured through channel descriptors residing in system memory. Before enabling a channel, the software must therefore take care to write this configuration to memory. When a channel is triggered, the DMA Controller will first read the channel descriptor from system memory, and then it will proceed to perform the memory transfers as specified by the descriptor. The descriptor contains the memory address to read from, the memory address to write to, the number of bytes to be transferred, etc. The channel descriptor is described in detail in Section H.4.3.

In addition to the basic transfer mode, the DMA Controller also supports two advanced transfer modes; ping-pong and scatter-gather. Ping-pong transfers are ideally suited for streaming data for high-speed peripheral communication as the DMA will be ready to retrieve the next incoming data bytes immediately while the processor core is still processing the previous ones (and similarly for outgoing communication). Scatter-gather involves executing a series of tasks from memory and allows sophisticated schemes to be implemented by software.

Using different priority levels for the channels and setting the number of bytes after which the DMA Controller re-arbitrates, it is possible to ensure that timing-critical transfers are serviced on time.

H.4.1 Channel Select Configuration

The channel select block allows selecting which peripheral's request lines (dma_req, dma_sreq) to connect to each DMA channel.

This configuration is done by software through the control registers DMA_CH0_CTRL-DMA_CH11_CTRL, with SOURCESEL and SIGSEL components. SOURCESEL selects which peripheral to listen to and SIGSEL picks which output signals to use from the selected peripheral.

All peripherals are connected to dma_req. When this signal is triggered, the DMA performs a number of transfers as specified by the channel descriptor (2^R). The USARTs are additionally connected to the dma_sreq line. When only dma_sreq is asserted but not dma_req, then the DMA will perform exactly one transfer only (given that dma_sreq is enabled by software).

H.4.2 DMA control

DMA arbitration rate

You can configure when the controller arbitrates during a DMA transfer. This enables you to reduce the latency to service a higher priority channel.

The controller provides four bits that configure how many AHB bus transfers occur before it re-arbitrates. These bits are known as the R_power bits because the value you enter, R, is raised to the power of two and this determines the arbitration rate. For example, if R = 4 then the arbitration rate is 2^4, that is, the controller arbitrates every 16 DMA transfers.

Table 46 lists the arbitration rates.

Figure 46: Arbitration rates

R_power	Arbitrate after x DMA transfers
b0000	x = 1
b0001	x = 2
b0010	x = 4
b0011	x = 8
b0100	x = 16
b0101	x = 32
b0110	x = 64
b0111	x = 128
b1000	x = 256
b1001	x = 512
b1010 - b1111	x = 1024

Note: You must take care not to assign a low-priority channel with a large R_power because this prevents the controller from servicing high-priority requests, until it re-arbitrates.

The number of dma transfers N that need to be done is specified by the user. When $N > 2^R$ and is not an integer multiple of 2^R then the controller always performs sequences of 2^R transfers until $N < 2^R$ remain to be transferred. The controller performs the remaining N transfers at the end of the DMA cycle.

You store the value of the R_power bits in the channel control data structure. See Section H.4.3 for more information about the location of the R_power bits in the data structure.

Priority

When the controller arbitrates, it determines the next channel to service by using the following information:

- the channel number
- the priority level, default or high, that is assigned to the channel.

You can configure each channel to use either the default priority level or a high priority level by setting the DMA_CHPRIS register.

Channel number zero has the highest priority and as the channel number increases, the priority of a channel decreases. Table 47 lists the DMA channel priority levels in descending order of priority.

Channel number	Priority level setting	Descending order of channel priority
0	High	Highest-priority DMA channel
1	High	-
2	High	-
3	High	-
4	High	-
5	High	-
6	High	-
7	High	-
8	High	-
9	High	-
10	High	-
11	High	-
0	Default	-
1	Default	-
2	Default	-
3	Default	-
4	Default	-
5	Default	-
6	Default	-
7	Default	-
8	Default	-
9	Default	-
10	Default	-
11	Default	Lowest-priority DMA channel

Figure 47: DMA channel priority levels

After a DMA transfer completes, the controller polls all the DMA channels that are available. Figure 48 shows the process it uses to determine which DMA transfer to perform next.

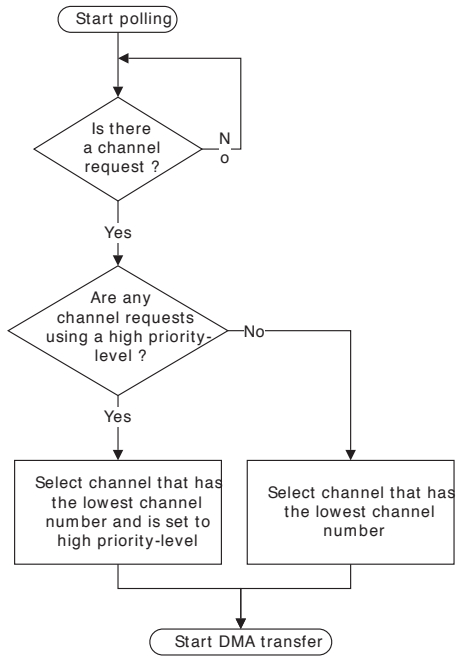


Figure 48: Polling flowchart

DMA cycle types

The cycle_ctrl bits control how the controller performs a DMA cycle. You can set the cycle_ctrl bits as Table 49 lists.

Note: The cycle_ctrl bits are located in the channel_cfg memory location that Section H.4.3 describes.

For all cycle types, the controller arbitrates after 2^R DMA transfers. If you set a low-priority channel with a large 2^R value then it prevents all other channels from performing a DMA transfer, until the low-priority DMA transfer completes. Therefore, you must take care when setting the R_power, that you do not significantly increase the latency for high-priority channels.

Invalid After the controller completes a DMA cycle it sets the cycle type to invalid, to prevent it from repeating the same DMA cycle.

cycle_ctrl	Description
b000	Channel control data structure is invalid
b001	Basic DMA transfer
b010	Auto-request
b011	Ping-pong
b100	Memory scatter-gather using the primary data structure
b101	Memory scatter-gather using the alternate data structure
b110	Peripheral scatter-gather using the primary data structure
b111	Peripheral scatter-gather using the alternate data structure

Figure 49: cycle_ctrl bits

Basic In this mode, you configure the controller to use either the primary or the alternate data structure. After you enable the channel C and the controller receives a request for this channel, then the flow for this DMA cycle is as follows:

1. The controller performs 2^R transfers. If the number of transfers remaining becomes zero, then the flow continues at step [?listitem].
2. The controller arbitrates:
   - if a higher-priority channel is requesting service then the controller services that channel
   - if the peripheral or software signals a request to the controller then it continues at step [?listitem].
3. The controller sets `dma_done[C]` HIGH for one `HFCORECLK` cycle. This indicates to the host processor that the DMA cycle is complete.

Auto-request When the controller operates in this mode, it is only necessary for it to receive a single request to enable it to complete the entire DMA cycle. This enables a large data transfer to occur, without significantly increasing the latency for servicing higher priority requests, or requiring multiple requests from the processor or peripheral.

You can configure the controller to use either the primary or the alternate data structure. After you enable the channel C and the controller receives a request for this channel, then the flow for this DMA cycle is as follows:

1. The controller performs 2^R transfers for channel C. If the number of transfers remaining is zero the flow continues at step [?listitem].

2. The controller arbitrates. When channel C has the highest priority then the DMA cycle continues at step [?listitem].

3. The controller sets `dma_done[C]` HIGH for one `HFCORECLK` cycle. This indicates to the host processor that the DMA cycle is complete.

Ping-pong In ping-pong mode, the controller performs a DMA cycle using one of the data structures (primary or alternate) and it then performs a DMA cycle using the other data structure. The controller continues to switch from primary to alternate to primary... until it reads a data structure that is invalid, or until the host processor disables the channel.

Figure 50 shows an example of a ping-pong DMA transaction.

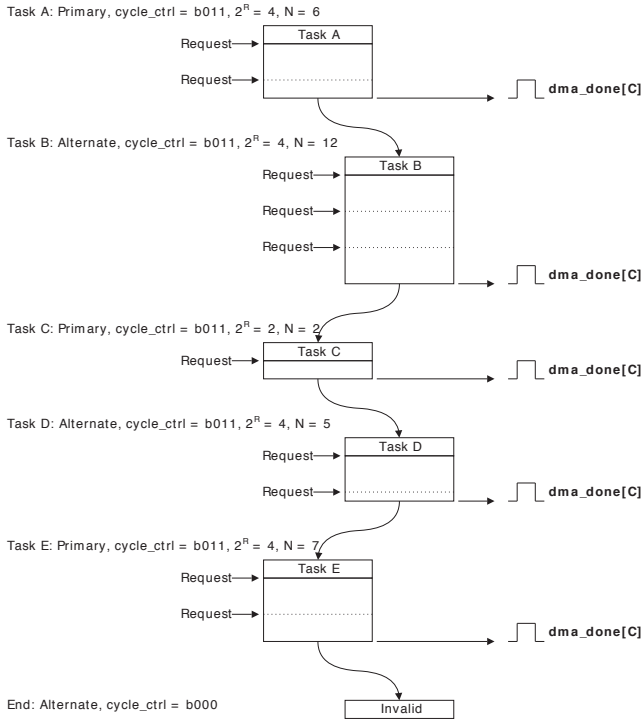


Figure 50: Ping-pong example

In Figure 50:

- Task A
  1. The host processor configures the primary data structure for task A.
  2. The host processor configures the alternate data structure for task B. This enables the controller to immediately switch to task B after task A completes, provided that a higher priority channel does not require servicing.

3. The controller receives a request and performs four DMA transfers.
4. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
5. The controller performs the remaining two DMA transfers.
6. The controller sets dma_done[C] HIGH for one HFCORECLK cycle and enters the arbitration process.

After task A completes, the host processor can configure the primary data structure for task C. This enables the controller to immediately switch to task C after task B completes, provided that a higher priority channel does not require servicing.

After the controller receives a new request for the channel and it has the highest priority then task B commences:

- Task B

4. The controller performs four DMA transfers.
5. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
6. The controller performs four DMA transfers.
7. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
8. The controller performs the remaining four DMA transfers.
9. The controller sets dma_done[C] HIGH for one HFCORECLK cycle and enters the arbitration process.

After task B completes, the host processor can configure the alternate data structure for task D.

After the controller receives a new request for the channel and it has the highest priority then task C commences:

- Task C

1. The controller performs two DMA transfers.
2. The controller sets dma_done[C] HIGH for one HFCORECLK cycle and enters the arbitration process.

After task C completes, the host processor can configure the primary data structure for task E.

After the controller receives a new request for the channel and it has the highest priority then task D commences:

- Task D

1. The controller performs four DMA transfers.
2. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
3. The controller performs the remaining DMA transfer.

4. The controller sets dma_done[C] HIGH for one HFCORECLK cycle and enters the arbitration process.

After the controller receives a new request for the channel and it has the highest priority then task E commences:

- Task E
  1. The controller performs four DMA transfers.
  2. The controller arbitrates. After the controller receives a request for this channel, the flow continues if the channel has the highest priority.
  3. The controller performs the remaining three DMA transfers.
  4. The controller sets dma_done[C] HIGH for one HFCORECLK cycle and enters the arbitration process.

If the controller receives a new request for the channel and it has the highest priority then it attempts to start the next task. However, because the host processor has not configured the alternate data structure, and on completion of task D the controller set the cycle_ctrl bits to b000, then the ping-pong DMA transaction completes.

Note: You can also terminate the ping-pong DMA cycle in Figure 50, if you configure task E to be a basic DMA cycle by setting the cycle_ctrl field to 3'b001.

Memory scatter-gather In memory scatter-gather mode the controller receives an initial request and then performs four DMA transfers using the primary data structure. After this transfer completes, it starts a DMA cycle using the alternate data structure. After this cycle completes, the controller performs another four DMA transfers using the primary data structure. The controller continues to switch from primary to alternate to primary... until either:

- the host processor configures the alternate data structure for a basic cycle
- it reads an invalid data structure.

  Note: After the controller completes the N primary transfers it invalidates the primary data structure by setting the cycle_ctrl field to b000.

The controller only asserts dma_done[C] when the scatter-gather transaction completes using an auto-request cycle.

In scatter-gather mode, the controller uses the primary data structure to program the alternate data structure. Table 51 lists the fields of the channel_cfg memory location for the primary data structure, that you must program with constant values and those that can be user defined.

[2] Because the R_power field is set to four, you must set N to be a multiple of four. The value given by N/4 is the number of times that you must configure the alternate data structure.

Bit	Field	Value	Description
Constant-value fields:			
[31:30]	dst_inc	b10	Configures the controller to use word increments for the address
[29:28]	dst_size	b10	Configures the controller to use word transfers
[27:26]	src_inc	b10	Configures the controller to use word increments for the address
[25:24]	src_size	b10	Configures the controller to use word transfers
17:14	R_power	b0010	Configures the controller to perform four DMA transfers
[3]	next_useburst	0	For a memory scatter-gather DMA cycle, this bit must be set to zero
[2:0]	cycle_ctrl	b100	Configures the controller to perform a memory scatter-gather DMA cycle
User defined values:			
[23:21]	dst_prot_ctrl	-	Configures the state of HPROT when the controller writes the destination data
[20:18]	src_prot_ctrl	-	Configures the state of HPROT when the controller reads the source data
[13:4]	n_minus_1	N^2	Configures the controller to perform N DMA transfers, where N is a multiple of four

Figure 51: Fields of channel_cfg memory location

Figure 52 shows a memory scatter-gather example.

Initialization:1. Configure primary to enable the copy A, B, C, and D operations: cycle_ctrl = b100, 2^R = 4, N = 16.
2. Write the primary source data to memory, using the structure shown in the following table.

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
Data for Task A	0x0A000000	0x0AE00000	cycle_ctrl = b101, 2^R = 4, N = 3	0xXXXXXXXX
Data for Task B	0x0B000000	0x0BE00000	cycle_ctrl = b101, 2^R = 2, N = 8	0xXXXXXXXX
Data for Task C	0x0C000000	0x0CE00000	cycle_ctrl = b101, 2^R = 8, N = 5	0xXXXXXXXX
Data for Task D	0x0D000000	0x0DE00000	cycle_ctrl = b010, 2^R = 4, N = 4	0xXXXXXXXX

Memory scatter-gather transaction:

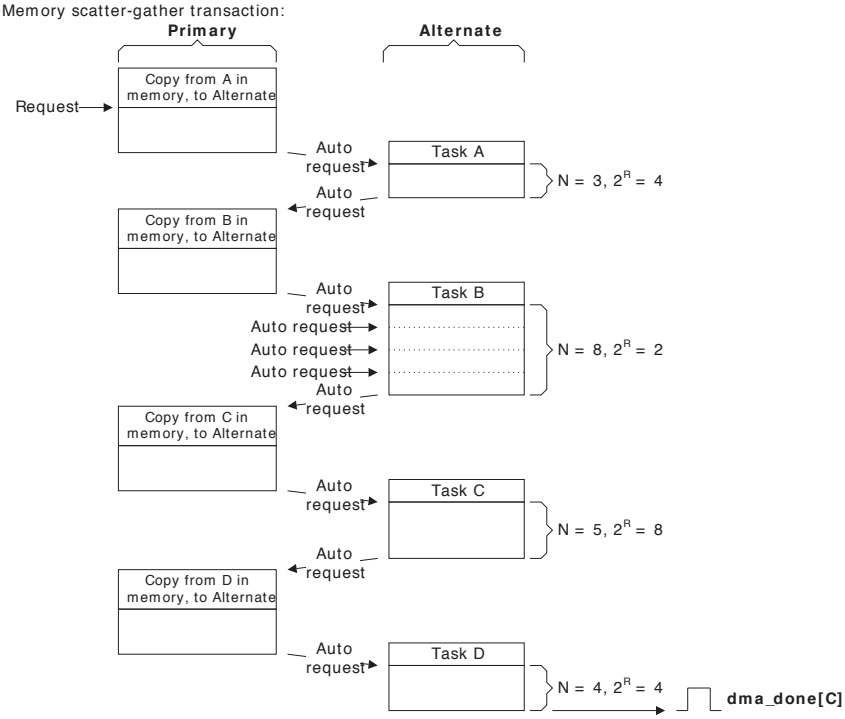


Figure 52: Memory scatter-gather example

In Figure 52:

1. The host processor configures the primary data structure to operate in memory scatter-gather mode by setting cycle_ctrl to b100. Because a data structure for a single channel consists of four words then you must set 2^R to 4. In this example, there are four tasks and therefore N is set to 16.
2. The host processor writes the data structure for tasks A, B, C, and D to the memory locations that the primary src_data_end_ptr specifies.
3. The host processor enables the channel.

The memory scatter-gather transaction commences when the controller receives a request on `dma_req[]` or a manual request from the host processor. The transaction continues as follows:

1. After receiving a request, the controller performs four DMA transfers. These transfers write the alternate data structure for task A.
2. The controller generates an auto-request for the channel and then arbitrates.
3. The controller performs task A. After it completes the task, it generates an auto-request for the channel and then arbitrates.
4. The controller performs four DMA transfers. These transfers write the alternate data structure for task B.
5. The controller generates an auto-request for the channel and then arbitrates.
6. The controller performs task B. After it completes the task, it generates an auto-request for the channel and then arbitrates.
7. The controller performs four DMA transfers. These transfers write the alternate data structure for task C.
8. The controller generates an auto-request for the channel and then arbitrates.
9. The controller performs task C. After it completes the task, it generates an auto-request for the channel and then arbitrates.
10. The controller performs four DMA transfers. These transfers write the alternate data structure for task D.
11. The controller sets the cycle_ctrl bits of the primary data structure to b000, to indicate that this data structure is now invalid.
12. The controller generates an auto-request for the channel and then arbitrates.
13. The controller performs task D using an auto-request cycle.
14. The controller sets `dma_done[C]` HIGH for one `HFCORECLK` cycle and enters the arbitration process.

Peripheral scatter-gather In peripheral scatter-gather mode the controller receives an initial request from a peripheral and then it performs four DMA transfers using the primary data structure. It then immediately starts a DMA cycle using the alternate data structure, without re-arbitrating.

Note: These are the only circumstances, where the controller does not enter the arbitration process after completing a transfer using the primary data structure.

After this cycle completes, the controller re-arbitrates and if the controller receives a request from the peripheral that has the highest priority then it performs another four DMA transfers using the primary data structure. It then immediately starts a DMA cycle using the alternate data structure, without re-arbitrating. The controller continues to switch from primary to alternate to primary... until either:

- the host processor configures the alternate data structure for a basic cycle

- it reads an invalid data structure.

  Note: After the controller completes the N primary transfers it invalidates the primary data structure by setting the cycle_ctrl field to b000.

The controller asserts `dma_done[C]` when the scatter-gather transaction completes using a basic cycle.

In scatter-gather mode, the controller uses the primary data structure to program the alternate data structure. Table 53 lists the fields of the channel_cfg memory location for the primary data structure, that you must program with constant values and those that can be user defined.

Figure 53: Fields of channel_cfg memory location

Bit	Field	Value	Description
Constant-value fields:			
[31:30]	dst_inc	b10	Configures the controller to use word increments for the address
[29:28]	dst_size	b10	Configures the controller to use word transfers
[27:26]	src_inc	b10	Configures the controller to use word increments for the address
[25:24]	src_size	b10	Configures the controller to use word transfers
17:14	R_power	b0010	Configures the controller to perform four DMA transfers
[2:0]	cycle_ctrl	b100	Configures the controller to perform a memory scatter-gather DMA cycle
User defined values:			
[23:21]	dst_prot_ctrl	-	Configures the state of `HPROT` when the controller writes the destination data
[20:18]	src_prot_ctrl	-	Configures the state of `HPROT` when the controller reads the source data
[13:4]	n_minus_1	N [3]	Configures the controller to perform N DMA transfers, where N is a multiple of four
[3]	next_useburst	-	When set to 1, the controller sets the chnl_useburst_set [C] bit to 1 after the alternate transfer completes

Figure 54 shows a peripheral scatter-gather example.

[3] Because the R_power field is set to four, you must set N to be a multiple of four. The value given by N/4 is the number of times that you must configure the alternate data structure.

Initialization:1. Configure primary to enable the copy A, B, C, and D operations: cycle_ctrl = b110, 2^R = 4, N = 16.
2. Write the primary source data in memory, using the structure shown in the following table.

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
Data for Task A	0x0A000000	0x0AE00000	cycle_ctrl = b111, 2^R = 4, N = 3	0xXXXXXXXX
Data for Task B	0x0B000000	0x0BE00000	cycle_ctrl = b111, 2^R = 2, N = 8	0xXXXXXXXX
Data for Task C	0x0C000000	0x0CE00000	cycle_ctrl = b111, 2^R = 8, N = 5	0xXXXXXXXX
Data for Task D	0x0D000000	0x0DE00000	cycle_ctrl = b001, 2^R = 4, N = 4	0xXXXXXXXX

Peripheral scatter-gather transaction:

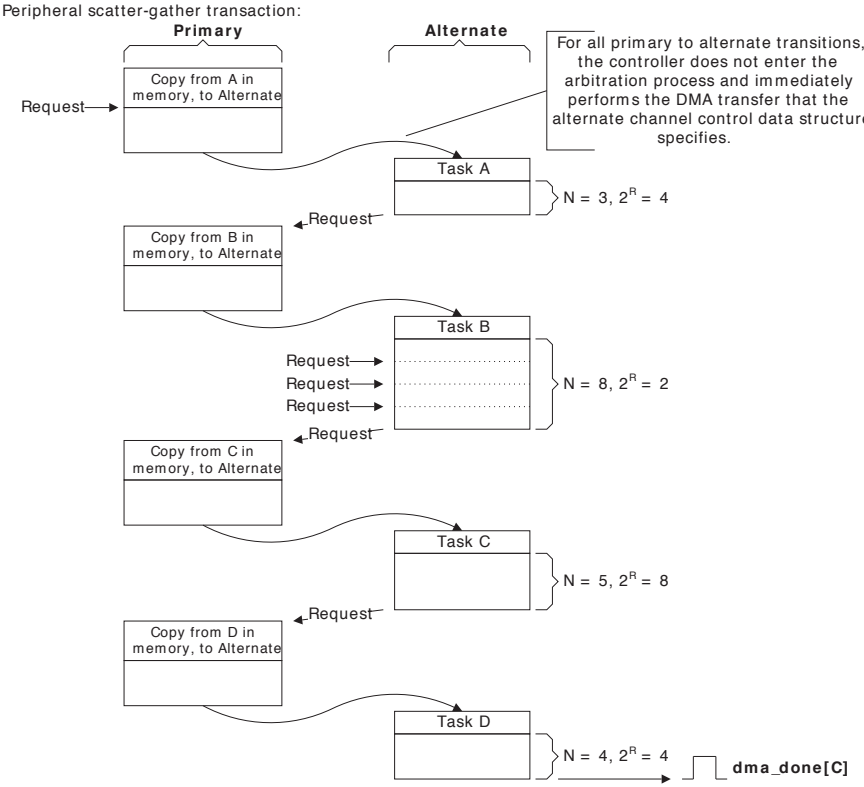


Figure 54: Peripheral scatter-gather example

In Figure 54:

1. The host processor configures the primary data structure to operate in peripheral scatter-gather mode by setting cycle_ctrl to b110. Because a data structure for a single channel consists of four words then you must set 2^R to 4. In this example, there are four tasks and therefore N is set to 16.
2. The host processor writes the data structure for tasks A, B, C, and D to the memory locations that the primary src_data_end_ptr specifies.
3. The host processor enables the channel.

The peripheral scatter-gather transaction commences when the controller receives a request on dma_req[]. The transaction continues as follows:

1. After receiving a request, the controller performs four DMA transfers. These transfers write the alternate data structure for task A.
2. The controller performs task A.
3. After the controller completes the task it enters the arbitration process.

After the peripheral issues a new request and it has the highest priority then the process continues with:

4. The controller performs four DMA transfers. These transfers write the alternate data structure for task B.
5. The controller performs task B. To enable the controller to complete the task, the peripheral must issue a further three requests.
6. After the controller completes the task it enters the arbitration process.

After the peripheral issues a new request and it has the highest priority then the process continues with:

4. The controller performs four DMA transfers. These transfers write the alternate data structure for task C.
5. The controller performs task C.
6. After the controller completes the task it enters the arbitration process.

After the peripheral issues a new request and it has the highest priority then the process continues with:

4. The controller performs four DMA transfers. These transfers write the alternate data structure for task D.
5. The controller sets the cycle_ctrl bits of the primary data structure to b000, to indicate that this data structure is now invalid.
6. The controller performs task D using a basic cycle.
7. The controller sets dma_done[C] HIGH for one HFCORECLK cycle and enters the arbitration process.

Error signaling

If the controller detects an ERROR response on the AHB-Lite master interface, it:

- disables the channel that corresponds to the ERROR
- sets dma_err HIGH.

After the host processor detects that dma_err is HIGH, it must check which channel was active when the ERROR occurred. It can do this by:

1. Reading the DMA_CHENS register to create a list of disabled channels.

When a channel asserts dma_done[] then the controller disables the channel. The program running on the host processor must always keep a record of which channels have recently asserted their dma_done[] outputs.

2. It must compare the disabled channels list from step [?listitem], with the record of the channels that have recently set their dma_done[] outputs. The channel with no record of dma_done[C] being set is the channel that the ERROR occurred on.

H.4.3 Channel control data structure

You must provide an area of system memory to contain the channel control data structure. This system memory must:

- provide a contiguous area of system memory that the controller and host processor can access
- have a base address that is an integer multiple of the total size of the channel control data structure.

Figure 55 shows the memory that the controller requires for the channel control data structure, when all 12 channels and the optional alternate data structure are in use.

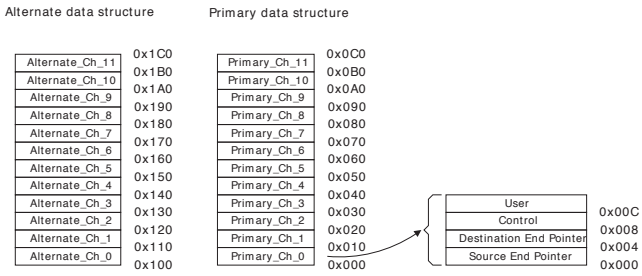


Figure 55: Memory map for 12 channels, including the alternate data structure

This structure in Figure 55 uses 384 bytes of system memory. The controller uses the lower 8 address bits to enable it to access all of the elements in the structure and therefore the base address must be at 0xXXXXXX00.

You can configure the base address for the primary data structure by writing the appropriate value in the DMA_CTRLBASE register.

You do not need to set aside the full 384 bytes if all dma channels are not used or if all alternate descriptors are not used. If, for example, only 4 channels are

used and they only need the primary descriptors, then only 64 bytes need to be set aside.

[?] lists the address bits that the controller uses when it accesses the elements of the channel control data structure.

Where:

A Selects one of the channel control data structures:

A = 0
Selects the primary data structure.

A = 1
Selects the alternate data structure.

C[3:0]
Selects the DMA channel.

Address[3:0]
Selects one of the control elements:

`0x0`
Selects the source data end pointer.

`0x4`
Selects the destination data end pointer.

`0x8`
Selects the control data configuration.

`0xC`
The controller does not access this address location. If required, you can enable the host processor to use this memory location as system memory.

Note: It is not necessary for you to calculate the base address of the alternate data structure because the DMA_ALTCTRLBASE register provides this information.

Figure 57 shows a detailed memory map of the descriptor structure.

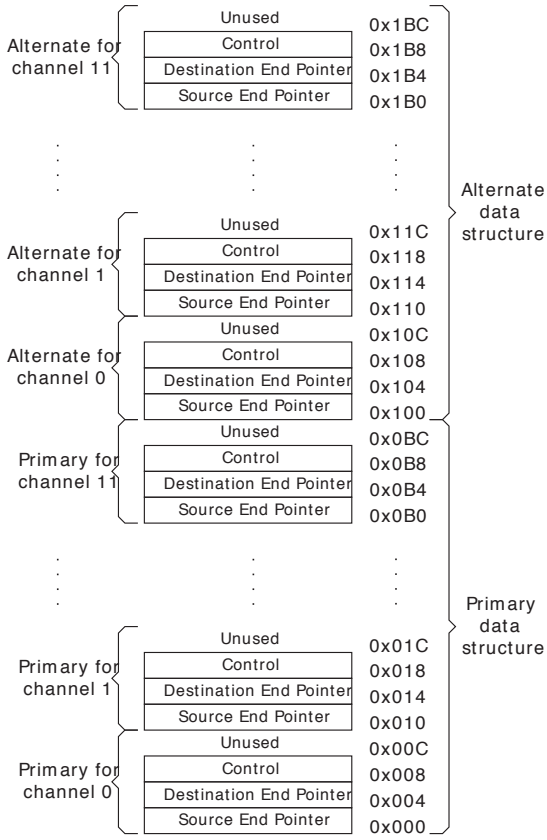


Figure 56: Detailed memory map for the 12 channels, including the alternate data structure

The controller uses the system memory to enable it to access two pointers and the control information that it requires for each channel. The following subsections will describe these 32-bit memory locations and how the controller calculates the DMA transfer address.

Source data end pointer

The src_data_end_ptr memory location contains a pointer to the end address of the source data. Figure 57 lists the bit assignments for this memory location.

Figure 57: Bit assignments

Bit	Name	Description
[31:0]	src_data_end_ptr	Pointer to the end address of the source data

Before the controller can perform a DMA transfer, you must program this memory location with the end address of the source data. The controller reads this memory location when it starts a 2^R DMA transfer.

Note: The controller does not write to this memory location.

Destination data end pointer

The dst_data_end_ptr memory location contains a pointer to the end address of the destination data. Table 58 lists the bit assignments for this memory location.

Figure 58: Bit assignments

Bit	Name	Description
[31:0]	dst_data_end_ptr	Pointer to the end address of the destination data

Before the controller can perform a DMA transfer, you must program this memory location with the end address of the destination data. The controller reads this memory location when it starts a 2^R DMA transfer.

Note: The controller does not write to this memory location.

Control data configuration

For each DMA transfer, the channel_cfg memory location provides the control information for the controller. Figure 59 shows the bit assignments for this memory location.

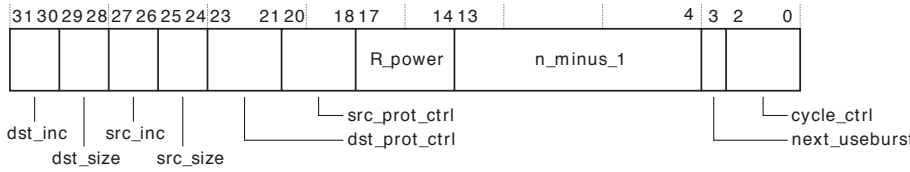


Figure 59: channel_cfg bit assignments

The following table lists the bit assignments for this memory location.

Bit	Name	Description
[31:30]	dst_inc	Destination address increment. The address increment depends on the source data width as follows: **Source data width = byte** b00 = byte b01 = halfword b10 = word b11 = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains. **Source data width = halfword** b00 = reserved b01 = halfword b10 = word b11 = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains. **Source data width = word** b00 = reserved b01 = reserved b10 = word b11 = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains.
[29:28]	dst_size	Destination data size. **Note:** You must set dst_size to contain the same value that src_size contains.
[27:26]	src_inc	Destination data size. Set the bits to control the source address increment. The address increment depends on the source data width as follows: **Source data width = byte** b00 = byte b01 = halfword b10 = word b11 = no increment. Address remains set to the value that the src_data_end_ptr memory location contains. **Source data width = halfword** b00 = reserved. b01 = halfword. b10 = word. b11 = no increment. Address remains set to the value that the src_data_end_ptr memory location contains. **Source data width = word** b00 = reserved. b01 = reserved. b10 = word. b11 = no increment. Address remains set to the value that the src_data_end_ptr memory location contains.

(continued)

Bit	Name	Description
[25:24]	src_size	Set the bits to match the size of the source data: b00 = byte b01 = halfword b10 = word b11 = reserved
[23:21]	dst_prot_ctrl	Set the bits to control the state of HPROT when the controller writes the destination data. **Bit23** This bit has no effect on the DMA. **Bit22** This bit has no effect on the DMA. **Bit21** Controls the state of HPROT as follows: 0 = HPROT is LOW and the access is non-privileged. 1 = HPROT is HIGH and the access is privileged.
[20:18]	src_prot_ctrl	Set the bits to control the state of HPROT when the controller reads the source data. **Bit20** This bit has no effect on the DMA. **Bit19** This bit has no effect on the DMA. **Bit18** Controls the state of HPROT as follows: 0 = HPROT is LOW and the access is non-privileged. 1 = HPROT is HIGH and the access is privileged.

(continued)

Bit	Name	Description
[17:14]	R_power	Set these bits to control how many DMA transfers can occur before the controller re-arbitrates. The possible arbitration rate settings are: **b0000** Arbitrates after each DMA transfer. **b0001** Arbitrates after 2 DMA transfers. **b0010** Arbitrates after 4 DMA transfers. **b0011** Arbitrates after 8 DMA transfers. **b0100** Arbitrates after 16 DMA transfers. **b0101** Arbitrates after 32 DMA transfers. **b0110** Arbitrates after 64 DMA transfers. **b0111** Arbitrates after 128 DMA transfers. **b1000** Arbitrates after 256 DMA transfers. **b1001** Arbitrates after 512 DMA transfers. **b1010-b1111** Arbitrates after 1024 DMA transfers. This means that no arbitration occurs during the DMA transfer because the maximum transfer size is 1024.
[13:4]	n_minus_1	Prior to the DMA cycle commencing, these bits represent the total number of DMA transfers that the DMA cycle contains. You must set these bits according to the size of DMA cycle that you require. The 10-bit value indicates the number of DMA transfers, minus one. The possible values are: b000000000 = 1 DMA transfer b000000001 = 2 DMA transfers b000000010 = 3 DMA transfers b000000011 = 4 DMA transfers b000000100 = 5 DMA transfers âĂę b111111111 = 1024 DMA transfers. The controller updates this field immediately prior to it entering the arbitration process. This enables the controller to store the number of outstanding DMA transfers that are necessary to complete the DMA cycle.

(continued)

Bit	Name	Description
[3]	next_useburst	Controls if the chnl_useburst_set [C] bit is set to a 1, when the controller is performing a peripheral scatter-gather and is completing a DMA cycle that uses the alternate data structure. **Note:** Immediately prior to completion of the DMA cycle that the alternate data structure specifies, the controller sets the chnl_useburst_set [C] bit to 0 if the number of remaining transfers is less than 2^R. The setting of the next_useburst bit controls if the controller performs an additional modification of the chnl_useburst_set [C] bit. In peripheral scatter-gather DMA cycle then after the DMA cycle that uses the alternate data structure completes , either: 0 = the controller does not change the value of the chnl_useburst_set [C] bit. If the chnl_useburst_set [C] bit is 0 then for all the remaining DMA cycles in the peripheral scatter-gather transaction, the controller responds to requests on `dma_req[]` and `dma_sreq[]`, when it performs a DMA cycle that uses an alternate data structure. 1 = the controller sets the chnl_useburst_set [C] bit to a 1. Therefore, for the remaining DMA cycles in the peripheral scatter-gather transaction, the controller only responds to requests on `dma_req[]`, when it performs a DMA cycle that uses an alternate data structure.

(continued)

Bit	Name	Description
[2:0]	cycle_ctrl	The operating mode of the DMA cycle. The modes are: **b000** Stop. Indicates that the data structure is invalid. **b001** Basic. The controller must receive a new request, prior to it entering the arbitration process, to enable the DMA cycle to complete. **b010** Auto-request. The controller automatically inserts a request for the appropriate channel during the arbitration process. This means that the initial request is sufficient to enable the DMA cycle to complete. **b011** Ping-pong. The controller performs a DMA cycle using one of the data structures. After the DMA cycle completes, it performs a DMA cycle using the other data structure. After the DMA cycle completes and provided that the host processor has updated the original data structure, it performs a DMA cycle using the original data structure. The controller continues to perform DMA cycles until it either reads an invalid data structure or the host processor changes the cycle_ctrl bits to b001 or b010. See Section H.4.2. **b100** Memory scatter/gather. See Section H.4.2. When the controller operates in memory scatter-gather mode, you must only use this value in the primary data structure. **b101** Memory scatter/gather. See Section H.4.2. When the controller operates in memory scatter-gather mode, you must only use this value in the alternate data structure. **b110** Peripheral scatter/gather. See Section H.4.2. When the controller operates in peripheral scatter-gather mode, you must only use this value in the primary data structure. **b111** Peripheral scatter/gather. See Section H.4.2. When the controller operates in peripheral scatter-gather mode, you must only use this value in the alternate data structure.

At the start of a DMA cycle, or 2^R DMA transfer, the controller fetches the channel_cfg from system memory. After it performs 2^R, or N, transfers it stores the updated channel_cfg in system memory.

The controller does not support a dst_size value that is different to the src_size value. If it detects a mismatch in these values, it uses the src_size value for source and destination and when it next updates the n_minus_1 field, it also sets the dst_size field to the same as the src_size field.

After the controller completes the N transfers it sets the cycle_ctrl field to b000, to indicate that the channel_cfg data is invalid. This prevents it from repeating the same DMA transfer.

Address calculation

To calculate the source address of a DMA transfer, the controller performs a left shift operation on the n_minus_1 value by a shift amount that src_inc specifies, and then subtracts the resulting value from the source data end pointer. Similarly, to calculate the destination address of a DMA transfer, it performs a left shift operation on the n_minus_1 value by a shift amount that dst_inc specifies, and then subtracts the resulting value from the destination end pointer.

Depending on the value of src_inc and dst_inc, the source address and destination address can be calculated using the equations:

- source address = src_data_end_ptr - n_minus_1
- destination address = dst_data_end_ptr - n_minus_1.
- source address = src_data_end_ptr - (n_minus_1 << 1)
- destination address = dst_data_end_ptr - (n_minus_1 << 1).
- source address = src_data_end_ptr - (n_minus_1 << 2)
- destination address = dst_data_end_ptr - (n_minus_1 << 2).
- source address = src_data_end_ptr
- destination address = dst_data_end_ptr.

Table 60 lists the destination addresses for a DMA cycle of six words.

Figure 60: Destination addresses for DMA cycle of six words

Initial values of channel_cfg, prior to the DMA cycle				
src_size = b10, dst_inc = b10, n_minus_1 = b101, cycle_ctrl = 1				
DMA transfers	End Pointer	Count	Difference [4]	Address
	0x2AC	5	0x14	0x298
	0x2AC	4	0x10	0x29C
	0x2AC	3	0xC	0x2A0
	0x2AC	2	0x8	0x2A4
	0x2AC	1	0x4	0x2A8
	0x2AC	0	0x0	0x2AC
Final values of channel_cfg, after the DMA cycle				
src_size = b10, dst_inc = b10, n_minus_1 = 0, cycle_ctrl = 0				

Table 61 lists the destination addresses for a DMA transfer of 12 bytes using a halfword increment.

[4] This value is the result of count being shifted left by the value of dst_inc.
[5] This value is the result of count being shifted left by the value of dst_inc.
[6] After the controller completes the DMA cycle it invalidates the channel_cfg memory location by clearing the cycle_ctrl field.

Initial values of channel_cfg, prior to the DMA cycle				
src_size = b00, dst_inc = b01, n_minus_1 = b1011, cycle_ctrl = 1, R_power = b11				
DMA transfers	End Pointer	Count	Difference [5]	Address
	0x5E7	11	0x16	0x5D1
	0x5E7	10	0x14	0x5D3
	0x5E7	9	0x12	0x5D5
	0x5E7	8	0x10	0x5D7
	0x5E7	7	0xE	0x5D9
	0x5E7	6	0xC	0x5DB
	0x5E7	5	0xA	0x5DD
	0x5E7	4	0x8	0x5DF
Values of channel_cfg after 2^R DMA transfers				
src_size = b00, dst_inc = b01, n_minus_1 = b011, cycle_ctrl = 1, R_power = b11				
DMA transfers	End Pointer	Count	Difference	Address
	0x5E7	3	0x6	0x5E1
	0x5E7	2	0x4	0x5E3
	0x5E7	1	0x2	0x5E5
	0x5E7	0	0x0	0x5E7
Final values of channel_cfg, after the DMA cycle				
src_size = b00, dst_inc = b01, n_minus_1 = 0, cycle_ctrl = 0 [6], R_power = b11				

Figure 61: Destination addresses for a DMA transfer of 12 bytes using halfword increment

H.4.4 Looped Transfers

A regular DMA channel is done when it has performed the number of transfers given by the channel descriptor. If an application wants a continuous flow of data, one option is to use ping-pong mode, alternating between two descriptors and having software update one descriptor while the other is being used. Another way is to use looped transfers.

For DMA channels 0 and 1, looping can be enabled by setting EN in DMA_LOOP0 and DMA_LOOP1 respectively. A looping DMA channel will on completion set the respective DONE interrupt flag, but then reload n_minus_1 in the channel descriptor with the loop width defined by WIDTH in DMA_LOOPx and continue transmitting data.

The total length of the transfer is given by the original value of n_minus_1 in the channel descriptor and WIDTH in DMA_LOOPx times the number of loops taken. The loop feature can for instance be used to implement a ring buffer, contiguously

overwriting old data when new data is available. To end the loop clear EN in DMA_LOOPx. The channel will then complete the last loop before stopping.

H.4.5 2D Copy

In addition to looped transfers, DMA channel 0 has the ability to do rectangle transfers, or 2D copy. 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. The DMA also has the ability to copy from linear data to a rectangle, and from a rectangle to linear data.

To set up rectangle copy for DMA channel 0, configure WIDTH in DMA_LOOP0 to one less than the rectangle width, and HEIGHT in DMA_RECT0 to one less than the rectangle height. Then set SRCSTRIDE in DMA_RECT0 to the outer rectangle width of the source, and DSTSTRIDE in DMA_RECT0 to the outer rectangle width of the destination rectangle. Finally, the channel descriptor for channel 0 has to be configured. The source and destination end pointers should be set to the last element of the first line of the source data and destination data respectively. The number of elements to be transferred, n_minus_1 should be set equal to WIDTH in DMA_LOOP0. The parameters are visualized in Figure 62.

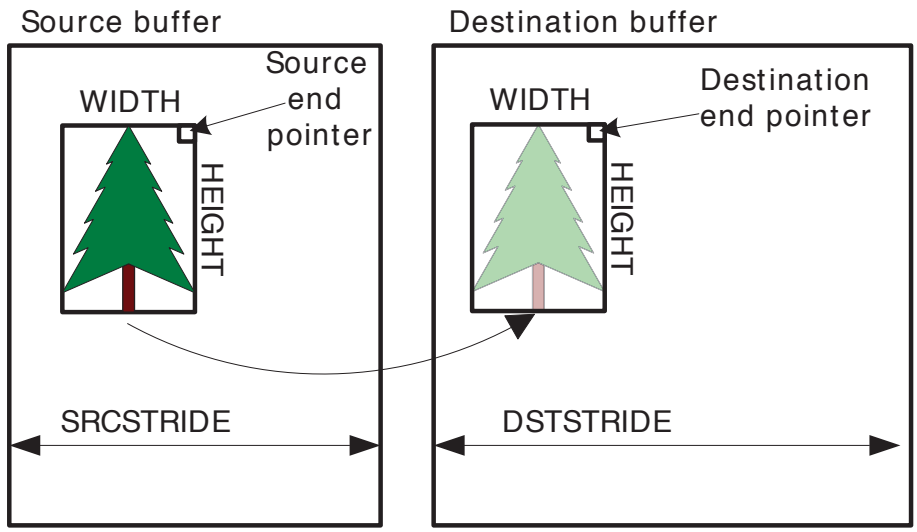


Figure 62: 2D copy

When doing a rectangle copy, the source and destination address of the channel descriptor will be incremented line for line as the DMA works its way through the rectangle. The operation is done when the number of lines specified by HEIGHT in DMA_RECT0 has been copied. The source and destination addresses in the channel descriptor will then point at the last element of the source and destination rectangles.

On completion, the DONE interrupt flag of channel 0 is set. Looping is not supported for rectangle copy.

In some cases, e.g. when performing graphics operations, it is desirable to create a list of copy operations and have them executed automatically. This can be done using 2D copy together with the scatter gather mode of the DMA controller. Set DESCRECT in DMA_CTRL to override SCRSTRIDE and HEIGHT in DMA_RECT0 and WIDTH in DMA_LOOP0 by the values in the user part of the DMA descriptor as shown in Table 63. In this way every copy command in the list can specify these parameters individually.

Figure 63: DMA descriptor

Bit	Field	Description
[30:20]	SRCSTRIDE	Stride in source buffer
[19:10]	HEIGHT	Height - 1 of data to be copied
[9:0]	WIDTH	Width - 1 of data to be copied

With regular 2D copy, the DMA descriptor will be updated as the copy operation proceeds. To be able to reuse 2D copy scatter gather list without rewriting source and destination end addresses, set PRDU in DMA_CTRL. This will prevent the address in the descriptor from being updated. In this case RDSCH0 in DMA_RDS must be set and all other bits in DMA_RDS must be cleared. The bits in DMA_RDS make individual DMA channels remember the source and destination end pointers while active, speeding up their transfers.

H.4.6 Interaction with the EMU

The DMA interacts with the Energy Management Unit (EMU) to allow transfers from , e.g., the LEUART to occur in EM2. The EMU can wake up the DMA sufficiently long to allow data transfers to occur. See section "DMA Support" in the LEUART documentation.

H.4.7 Interrupts

The PL230 dma_done[n:0] signals (one for each channel) as well as the dma_err signal, are available as interrupts to the Cortex-M3 core. They are combined into one interrupt vector, DMA_INT. If the interrupt for the DMA is enabled in the ARM Cortex-M3 core, an interrupt will be made if one or more of the interrupt flags in DMA_IF and their corresponding bits in DMA_IEN are set.

H.5 Examples

A basic example of how to program the DMA for transferring 42 bytes from the USART1 to memory location 0x20003420. Assumes that the channel 0 is currently disabled, and that the DMA_ALTCTRLBASE register has already been configured.

DMA Transfer

1. Configure the channel select for using USART1 with DMA channel 0
   1. Write SOURCESEL=0b001101 and SIGSEL=XX to DMA_CHCTRL0
2. Configure the primary channel descriptor for DMA channel 0
   1. Write XX (read address of USART1) to src_data_end_ptr
   2. Write 0x20003420 + 40 to dst_data_end_ptr c
   3. Write these values to channel_cfg for channel 0:
      1. dst_inc=b01 (destination halfword address increment)
      2. dst_size=b01 (halfword transfer size)
      3. src_inc=b11 (no address increment for source)
      4. src_size=01 (halfword transfer size)
      5. dst_prot_ctrl=000 (no cache/buffer/privilege)
      6. src_prot_ctrl=000 (no cache/buffer/privilege)
      7. R_power=b0000 (arbitrate after each DMA transfer)
      8. n_minus_1=d20 (transfer 21 halfwords)
      9. next_useburst=b0 (not applicable)
      10. cycle_ctrl=b001 (basic operating mode)
3. Enable the DMA
   1. Write EN=1 to DMA_CONFIG
4. Disable the single requests for channel 0 (i.e., do not react to data available, wait for buffer full)
   1. Write DMA_CHUSEBURSTS[0]=1
5. Enable buffer-full requests for channel 0
   1. Write DMA_CHREQMASKC[0]=1
6. Use the primary data structure for channel 0
   1. Write DMA_CHALTC[0]=1
7. Enable channel 0
   1. Write DMA_CHENS[0]=1

H.6 Register Map

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

Offset	Name	Type	Description
0x000	DMA_STATUS	R	DMA Status Registers
0x004	DMA_CONFIG	W	DMA Configuration Register
0x008	DMA_CTRLBASE	RW	Channel Control Data Base Pointer Register
0x00C	DMA_ALTCTRLBASE	R	Channel Alternate Control Data Base Pointer Register
0x010	DMA_CHWAITSTATUS	R	Channel Wait on Request Status Register
0x014	DMA_CHSWREQ	W1	Channel Software Request Register
0x018	DMA_CHUSEBURSTS	RW1H	Channel Useburst Set Register
0x01C	DMA_CHUSEBURSTC	W1	Channel Useburst Clear Register
0x020	DMA_CHREQMASKS	RW1	Channel Request Mask Set Register
0x024	DMA_CHREQMASKC	W1	Channel Request Mask Clear Register
0x028	DMA_CHENS	RW1	Channel Enable Set Register
0x02C	DMA_CHENC	W1	Channel Enable Clear Register
0x030	DMA_CHALTS	RW1	Channel Alternate Set Register
0x034	DMA_CHALTC	W1	Channel Alternate Clear Register
0x038	DMA_CHPRIS	RW1	Channel Priority Set Register
0x03C	DMA_CHPRIC	W1	Channel Priority Clear Register
0x04C	DMA_ERRORC	RW	Bus Error Clear Register
0xE10	DMA_CHREQSTATUS	R	Channel Request Status
0xE18	DMA_CHSREQSTATUS	R	Channel Single Request Status
0x1000	DMA_IF	R	Interrupt Flag Register
0x1004	DMA_IFS	W1	Interrupt Flag Set Register
0x1008	DMA_IFC	W1	Interrupt Flag Clear Register
0x100C	DMA_IEN	RW	Interrupt Enable register
0x1010	DMA_CTRL	RW	DMA Control Register
0x1014	DMA_RDS	RW	DMA Retain Descriptor State
0x1020	DMA_LOOP0	RWH	Channel 0 Loop Register
0x1024	DMA_LOOP1	RW	Channel 1 Loop Register
0x1060	DMA_RECT0	RWH	Channel 0 Rectangle Register

Offset	Name	Type	Description
0x1100	DMA_CH0_CTRL	RW	Channel Control Register
...	DMA_CHx_CTRL	RW	Channel Control Register
0x112C	DMA_CH11_CTRL	RW	Channel Control Register

H.7 Register Description

H.7.1 DMA_STATUS - DMA Status 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												0x0B													0x0							0
Access												R													R							R
Name												CHNUM													STATE							EN

Bit	Name	Reset	Access	Description
31:21	Reserved	To ensure compatibility with future devices, always write bits to 0.		
20:16	CHNUM	0x0B	R	**Channel Number**
	Number of available DMA channels minus one.			
15:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:4	STATE	0x0	R	**Control Current State**
	State can be one of the following. Higher values (11-15) are undefined.			

Value	Mode	Description
0	IDLE	Idle
1	RDCHCTRLDATA	Reading channel controller data
2	RDSRCENDPTR	Reading source data end pointer
3	RDDSTENDPTR	Reading destination data end pointer
4	RDSRCDATA	Reading source data
5	WRDSTDATA	Writing destination data
6	WAITREQCLR	Waiting for DMA request to clear
7	WRCHCTRLDATA	Writing channel controller data
8	STALLED	Stalled
9	DONE	Done
10	PERSCATTRANS	Peripheral scatter-gather transition

3:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	EN	0	R	**DMA Enable Status**
	When this bit is 1, the DMA is enabled.			

H.7.2 DMA_CONFIG - DMA 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																											0					0
Access																											W					W
Name																											CHPROT					EN

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5	CHPROT	0	W	**Channel Protection Control**
	Control whether accesses done by the DMA controller are privileged or not. When CHPROT = 1 then HPROT is HIGH and the access is privileged. When CHPROT = 0 then HPROT is LOW and the access is non-privileged.			
4:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	EN	0	W	**Enable DMA**
	Set this bit to enable the DMA controller.			

H.7.3 DMA_CTRLBASE - Channel Control Data Base Pointer 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	CTRLBASE																															

Bit	Name	Reset	Access	Description
31:0	CTRLBASE	0x00000000	RW	**Channel Control Data Base Pointer**
	The base pointer for a location in system memory that holds the channel control data structure. This register must be written to point to a location in system memory with the channel control data structure before the DMA can be used. Note that ctrl_base_ptr[8:0] must be 0.			

H.7.4 DMA_ALTCTRLBASE - Channel Alternate Control Data Base Pointer 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	0x00000100																															
Access	R																															
Name	ALTCTRLBASE																															

Bit	Name	Reset	Access	Description
31:0	ALTCTRLBASE	0x00000100	R	**Channel Alternate Control Data Base Pointer**
	The base address of the alternate data structure. This register will read as DMA_CTRLBASE + 0x100.			

H.7.5 DMA_CHWAITSTATUS - Channel Wait on Request 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																					1	1	1	1	1	1	1	1	1	1	1	1
Access																					R	R	R	R	R	R	R	R	R	R	R	R

Offset	Bit Position																							
Name																	CH11WAITSTATUS	CH10WAITSTATUS	CH9WAITSTATUS	CH8WAITSTATUS	CH7WAITSTATUS	CH6WAITSTATUS	CH5WAITSTATUS	CH4WAITSTATUS	CH3WAITSTATUS	CH2WAITSTATUS	CH1WAITSTATUS	CH0WAITSTATUS

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11WAITSTATUS	1	R	**Channel 11 Wait on Request Status**
	Status for wait on request for channel 11.			
10	CH10WAITSTATUS	1	R	**Channel 10 Wait on Request Status**
	Status for wait on request for channel 10.			
9	CH9WAITSTATUS	1	R	**Channel 9 Wait on Request Status**
	Status for wait on request for channel 9.			
8	CH8WAITSTATUS	1	R	**Channel 8 Wait on Request Status**
	Status for wait on request for channel 8.			
7	CH7WAITSTATUS	1	R	**Channel 7 Wait on Request Status**
	Status for wait on request for channel 7.			
6	CH6WAITSTATUS	1	R	**Channel 6 Wait on Request Status**
	Status for wait on request for channel 6.			
5	CH5WAITSTATUS	1	R	**Channel 5 Wait on Request Status**
	Status for wait on request for channel 5.			
4	CH4WAITSTATUS	1	R	**Channel 4 Wait on Request Status**
	Status for wait on request for channel 4.			
3	CH3WAITSTATUS	1	R	**Channel 3 Wait on Request Status**
	Status for wait on request for channel 3.			
2	CH2WAITSTATUS	1	R	**Channel 2 Wait on Request Status**
	Status for wait on request for channel 2.			
1	CH1WAITSTATUS	1	R	**Channel 1 Wait on Request Status**
	Status for wait on request for channel 1.			
0	CH0WAITSTATUS	1	R	**Channel 0 Wait on Request Status**
	Status for wait on request for channel 0.			

H.7.6 DMA_CHSWREQ - Channel Software Request 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
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																					CH11SWREQ	CH10SWREQ	CH9SWREQ	CH8SWREQ	CH7SWREQ	CH6SWREQ	CH5SWREQ	CH4SWREQ	CH3SWREQ	CH2SWREQ	CH1SWREQ	CH0SWREQ

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11SWREQ	0	W1	**Channel 11 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
10	CH10SWREQ	0	W1	**Channel 10 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			

Bit	Name	Reset	Access	Description
9	CH9SWREQ	0	W1	**Channel 9 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
8	CH8SWREQ	0	W1	**Channel 8 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
7	CH7SWREQ	0	W1	**Channel 7 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
6	CH6SWREQ	0	W1	**Channel 6 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
5	CH5SWREQ	0	W1	**Channel 5 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
4	CH4SWREQ	0	W1	**Channel 4 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
3	CH3SWREQ	0	W1	**Channel 3 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
2	CH2SWREQ	0	W1	**Channel 2 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
1	CH1SWREQ	0	W1	**Channel 1 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			
0	CH0SWREQ	0	W1	**Channel 0 Software Request**
	Write 1 to this bit to generate a DMA request for this channel.			

H.7.7 DMA_CHUSEBURSTS - Channel Useburst 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
Access																					RW1H	RW1H	RW1H	RW1H	RW1H	RW1H	RW1H	RW1H	RW1H	RW1H	RW1H	RW1H
Name																					CH11USEBURSTS	CH10USEBURSTS	CH9USEBURSTS	CH8USEBURSTS	CH7USEBURSTS	CH6USEBURSTS	CH5USEBURSTS	CH4USEBURSTS	CH3USEBURSTS	CH2USEBURSTS	CH1USEBURSTS	CH0USEBURSTS

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11USEBURSTS	0	RW1H	**Channel 11 Useburst Set**
	See description for channel 0.			
10	CH10USEBURSTS	0	RW1H	**Channel 10 Useburst Set**
	See description for channel 0.			
9	CH9USEBURSTS	0	RW1H	**Channel 9 Useburst Set**
	See description for channel 0.			
8	CH8USEBURSTS	0	RW1H	**Channel 8 Useburst Set**
	See description for channel 0.			
7	CH7USEBURSTS	0	RW1H	**Channel 7 Useburst Set**
	See description for channel 0.			
6	CH6USEBURSTS	0	RW1H	**Channel 6 Useburst Set**
	See description for channel 0.			
5	CH5USEBURSTS	0	RW1H	**Channel 5 Useburst Set**
	See description for channel 0.			
4	CH4USEBURSTS	0	RW1H	**Channel 4 Useburst Set**
	See description for channel 0.			
3	CH3USEBURSTS	0	RW1H	**Channel 3 Useburst Set**

Bit	Name	Reset	Access	Description
	See description for channel 0.			
2	CH2USEBURSTS	0	RW1H	**Channel 2 Useburst Set**
	See description for channel 0.			
1	CH1USEBURSTS	0	RW1H	**Channel 1 Useburst Set**
	See description for channel 0.			
0	CH0USEBURSTS	0	RW1H	**Channel 0 Useburst Set**

Write to 1 to enable the useburst setting for this channel. Reading returns the useburst status. After the penultimate 2^R transfer completes, if the number of remaining transfers, N, is less than 2^R then the controller resets the chnl_useburst_set bit to 0. This enables you to complete the remaining transfers using dma_req[] or dma_sreq[]. In peripheral scatter-gather mode, if the next_useburst bit is set in channel_cfg then the controller sets the chnl_useburst_set[C] bit to a 1, when it completes the DMA cycle that uses the alternate data structure.

Value	Mode	Description
0	SINGLEANDBURST	Channel responds to both single and burst requests
1	BURSTONLY	Channel responds to burst requests only

H.7.8 DMA_CHUSEBURSTC - Channel Useburst 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
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																					CH11USEBURSTC	CH10USEBURSTC	CH9USEBURSTC	CH08USEBURSTC	CH7USEBURSTC	CH6USEBURSTC	CH5USEBURSTC	CH4USEBURSTC	CH3USEBURSTC	CH2USEBURSTC	CH1USEBURSTC	CH0USEBURSTC

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11USEBURSTC	0	W1	**Channel 11 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
10	CH10USEBURSTC	0	W1	**Channel 10 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
9	CH9USEBURSTC	0	W1	**Channel 9 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
8	CH08USEBURSTC	0	W1	**Channel 8 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
7	CH7USEBURSTC	0	W1	**Channel 7 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
6	CH6USEBURSTC	0	W1	**Channel 6 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
5	CH5USEBURSTC	0	W1	**Channel 5 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
4	CH4USEBURSTC	0	W1	**Channel 4 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
3	CH3USEBURSTC	0	W1	**Channel 3 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
2	CH2USEBURSTC	0	W1	**Channel 2 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
1	CH1USEBURSTC	0	W1	**Channel 1 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			
0	CH0USEBURSTC	0	W1	**Channel 0 Useburst Clear**
	Write to 1 to disable useburst setting for this channel.			

H.7.9 DMA_CHREQMASKS - Channel Request Mask 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	0	0	0	0	0	0	0
Access																					RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																					CH11REQMASKS	CH10REQMASKS	CH9REQMASKS	CH8REQMASKS	CH7REQMASKS	CH6REQMASKS	CH5REQMASKS	CH4REQMASKS	CH3REQMASKS	CH2REQMASKS	CH1REQMASKS	CH0REQMASKS

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11REQMASKS	0	RW1	**Channel 11 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
10	CH10REQMASKS	0	RW1	**Channel 10 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
9	CH9REQMASKS	0	RW1	**Channel 9 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
8	CH8REQMASKS	0	RW1	**Channel 8 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
7	CH7REQMASKS	0	RW1	**Channel 7 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
6	CH6REQMASKS	0	RW1	**Channel 6 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
5	CH5REQMASKS	0	RW1	**Channel 5 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
4	CH4REQMASKS	0	RW1	**Channel 4 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
3	CH3REQMASKS	0	RW1	**Channel 3 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
2	CH2REQMASKS	0	RW1	**Channel 2 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
1	CH1REQMASKS	0	RW1	**Channel 1 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			
0	CH0REQMASKS	0	RW1	**Channel 0 Request Mask Set**
	Write to 1 to disable peripheral requests for this channel.			

H.7.10 DMA_CHREQMASKC - Channel Request Mask 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	0	0	0	0	0	0	0
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																					CH11REQMASKC	CH10REQMASKC	CH9REQMASKC	CH8REQMASKC	CH7REQMASKC	CH6REQMASKC	CH5REQMASKC	CH4REQMASKC	CH3REQMASKC	CH2REQMASKC	CH1REQMASKC	CH0REQMASKC

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11REQMASKC	0	W1	**Channel 11 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
10	CH10REQMASKC	0	W1	**Channel 10 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
9	CH9REQMASKC	0	W1	**Channel 9 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
8	CH8REQMASKC	0	W1	**Channel 8 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
7	CH7REQMASKC	0	W1	**Channel 7 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
6	CH6REQMASKC	0	W1	**Channel 6 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
5	CH5REQMASKC	0	W1	**Channel 5 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
4	CH4REQMASKC	0	W1	**Channel 4 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
3	CH3REQMASKC	0	W1	**Channel 3 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
2	CH2REQMASKC	0	W1	**Channel 2 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
1	CH1REQMASKC	0	W1	**Channel 1 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			
0	CH0REQMASKC	0	W1	**Channel 0 Request Mask Clear**
	Write to 1 to enable peripheral requests for this channel.			

H.7.11 DMA_CHENS - Channel Enable 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
Access																					RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																					CH11ENS	CH10ENS	CH9ENS	CH8ENS	CH7ENS	CH6ENS	CH5ENS	CH4ENS	CH3ENS	CH2ENS	CH1ENS	CH0ENS

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11ENS	0	RW1	**Channel 11 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
10	CH10ENS	0	RW1	**Channel 10 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
9	CH9ENS	0	RW1	**Channel 9 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
8	CH8ENS	0	RW1	**Channel 8 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
7	CH7ENS	0	RW1	**Channel 7 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
6	CH6ENS	0	RW1	**Channel 6 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
5	CH5ENS	0	RW1	**Channel 5 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			

Bit	Name	Reset	Access	Description
4	CH4ENS	0	RW1	**Channel 4 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
3	CH3ENS	0	RW1	**Channel 3 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
2	CH2ENS	0	RW1	**Channel 2 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
1	CH1ENS	0	RW1	**Channel 1 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			
0	CH0ENS	0	RW1	**Channel 0 Enable Set**
	Write to 1 to enable this channel. Reading returns the enable status of the channel.			

H.7.12 DMA_CHENC - Channel Enable 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
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																					CH11ENC	CH10ENC	CH9ENC	CH8ENC	CH7ENC	CH6ENC	CH5ENC	CH4ENC	CH3ENC	CH2ENC	CH1ENC	CH0ENC

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11ENC	0	W1	**Channel 11 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
10	CH10ENC	0	W1	**Channel 10 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
9	CH9ENC	0	W1	**Channel 9 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
8	CH8ENC	0	W1	**Channel 8 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
7	CH7ENC	0	W1	**Channel 7 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
6	CH6ENC	0	W1	**Channel 6 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
5	CH5ENC	0	W1	**Channel 5 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
4	CH4ENC	0	W1	**Channel 4 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
3	CH3ENC	0	W1	**Channel 3 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
2	CH2ENC	0	W1	**Channel 2 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
1	CH1ENC	0	W1	**Channel 1 Enable Clear**
	Write to 1 to disable this channel. See also description for channel 0.			
0	CH0ENC	0	W1	**Channel 0 Enable Clear**
	Write to 1 to disable this channel. Note that the controller disables a channel, by setting the appropriate bit, when either it completes the DMA cycle, or it reads a channel_cfg memory location which has cycle_ctrl = b000, or an ERROR occurs on the AHB-Lite bus. A read from this field returns the value of CH0ENS from the DMA_CHENS register.			

H.7.13 DMA_CHALTS - Channel Alternate 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	0	0	0
Access																					RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																					CH11ALTS	CH10ALTS	CH9ALTS	CH8ALTS	CH7ALTS	CH6ALTS	CH5ALTS	CH4ALTS	CH3ALTS	CH2ALTS	CH1ALTS	CH0ALTS

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11ALTS	0	RW1	**Channel 11 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
10	CH10ALTS	0	RW1	**Channel 10 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
9	CH9ALTS	0	RW1	**Channel 9 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
8	CH8ALTS	0	RW1	**Channel 8 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
7	CH7ALTS	0	RW1	**Channel 7 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
6	CH6ALTS	0	RW1	**Channel 6 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
5	CH5ALTS	0	RW1	**Channel 5 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
4	CH4ALTS	0	RW1	**Channel 4 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
3	CH3ALTS	0	RW1	**Channel 3 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
2	CH2ALTS	0	RW1	**Channel 2 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
1	CH1ALTS	0	RW1	**Channel 1 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			
0	CH0ALTS	0	RW1	**Channel 0 Alternate Structure Set**
	Write to 1 to select the alternate structure for this channel.			

H.7.14 DMA_CHALTC - Channel Alternate 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	0	0	0
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																					CH11ALTC	CH10ALTC	CH9ALTC	CH8ALTC	CH7ALTC	CH6ALTC	CH5ALTC	CH4ALTC	CH3ALTC	CH2ALTC	CH1ALTC	CH0ALTC

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11ALTC	0	W1	**Channel 11 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
10	CH10ALTC	0	W1	**Channel 10 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			

Bit	Name	Reset	Access	Description
9	CH9ALTC	0	W1	**Channel 9 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
8	CH8ALTC	0	W1	**Channel 8 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
7	CH7ALTC	0	W1	**Channel 7 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
6	CH6ALTC	0	W1	**Channel 6 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
5	CH5ALTC	0	W1	**Channel 5 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
4	CH4ALTC	0	W1	**Channel 4 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
3	CH3ALTC	0	W1	**Channel 3 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
2	CH2ALTC	0	W1	**Channel 2 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
1	CH1ALTC	0	W1	**Channel 1 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			
0	CH0ALTC	0	W1	**Channel 0 Alternate Clear**
	Write to 1 to select the primary structure for this channel.			

H.7.15 DMA_CHPRIS - Channel Priority 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
Access																					RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1	RW1
Name																					CH11PRIS	CH10PRIS	CH9PRIS	CH8PRIS	CH7PRIS	CH6PRIS	CH5PRIS	CH4PRIS	CH3PRIS	CH2PRIS	CH1PRIS	CH0PRIS

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11PRIS	0	RW1	**Channel 11 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
10	CH10PRIS	0	RW1	**Channel 10 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
9	CH9PRIS	0	RW1	**Channel 9 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
8	CH8PRIS	0	RW1	**Channel 8 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
7	CH7PRIS	0	RW1	**Channel 7 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
6	CH6PRIS	0	RW1	**Channel 6 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
5	CH5PRIS	0	RW1	**Channel 5 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
4	CH4PRIS	0	RW1	**Channel 4 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
3	CH3PRIS	0	RW1	**Channel 3 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
2	CH2PRIS	0	RW1	**Channel 2 High Priority Set**

Bit	Name	Reset	Access	Description
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
1	CH1PRIS	0	RW1	**Channel 1 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			
0	CH0PRIS	0	RW1	**Channel 0 High Priority Set**
	Write to 1 to obtain high priority for this channel. Reading returns the channel priority status.			

H.7.16 DMA_CHPRIC - Channel Priority 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
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																					CH11PRIC	CH10PRIC	CH9PRIC	CH8PRIC	CH7PRIC	CH6PRIC	CH5PRIC	CH4PRIC	CH3PRIC	CH2PRIC	CH1PRIC	CH0PRIC

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11PRIC	0	W1	**Channel 11 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
10	CH10PRIC	0	W1	**Channel 10 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
9	CH9PRIC	0	W1	**Channel 9 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
8	CH8PRIC	0	W1	**Channel 8 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
7	CH7PRIC	0	W1	**Channel 7 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
6	CH6PRIC	0	W1	**Channel 6 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
5	CH5PRIC	0	W1	**Channel 5 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
4	CH4PRIC	0	W1	**Channel 4 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
3	CH3PRIC	0	W1	**Channel 3 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
2	CH2PRIC	0	W1	**Channel 2 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
1	CH1PRIC	0	W1	**Channel 1 High Priority Clear**
	Write to 1 to clear high priority for this channel.			
0	CH0PRIC	0	W1	**Channel 0 High Priority Clear**
	Write to 1 to clear high priority for this channel.			

H.7.17 DMA_ERRORC - Bus Error Clear Register

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

Offset	Bit Position	
Name		ERRORC

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	ERRORC	0	RW	**Bus Error Clear**
	This bit is set high if an AHB bus error has occurred. Writing a 1 to this bit will clear the bit. If the error is deasserted at the same time as an error occurs on the bus, the error condition takes precedence and ERRORC remains asserted.			

H.7.18 DMA_CHREQSTATUS - Channel Request Status

Offset	Bit Position																											
0xE10	31	30	29	28	27	26	25	24	23	22	21	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	R	R	R	R	R	R	R	R	R	R	R
Name																					CH11REQSTATUS	CH10REQSTATUS	CH9REQSTATUS	CH8REQSTATUS	CH7REQSTATUS	CH6REQSTATUS	CH5REQSTATUS	CH4REQSTATUS	CH3REQSTATUS	CH2REQSTATUS	CH1REQSTATUS	CH0REQSTATUS

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11REQSTATUS	0	R	**Channel 11 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
10	CH10REQSTATUS	0	R	**Channel 10 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
9	CH9REQSTATUS	0	R	**Channel 9 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
8	CH8REQSTATUS	0	R	**Channel 8 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
7	CH7REQSTATUS	0	R	**Channel 7 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
6	CH6REQSTATUS	0	R	**Channel 6 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
5	CH5REQSTATUS	0	R	**Channel 5 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
4	CH4REQSTATUS	0	R	**Channel 4 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
3	CH3REQSTATUS	0	R	**Channel 3 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
2	CH2REQSTATUS	0	R	**Channel 2 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			

Bit	Name	Reset	Access	Description
1	CH1REQSTATUS	0	R	**Channel 1 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			
0	CH0REQSTATUS	0	R	**Channel 0 Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using 2^R DMA transfers.			

H.7.19 DMA_CHSREQSTATUS - Channel Single Request Status

Offset	Bit Position																											
0xE18	31	30	29	28	27	26	25	24	23	22	21	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	R	R	R	R	R	R	R	R	R	R	R
Name																					CH11SREQSTATUS	CH10SREQSTATUS	CH9SREQSTATUS	CH8SREQSTATUS	CH7SREQSTATUS	CH6SREQSTATUS	CH5SREQSTATUS	CH4SREQSTATUS	CH3SREQSTATUS	CH2SREQSTATUS	CH1SREQSTATUS	CH0SREQSTATUS

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11SREQSTATUS	0	R	**Channel 11 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
10	CH10SREQSTATUS	0	R	**Channel 10 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
9	CH9SREQSTATUS	0	R	**Channel 9 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
8	CH8SREQSTATUS	0	R	**Channel 8 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
7	CH7SREQSTATUS	0	R	**Channel 7 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
6	CH6SREQSTATUS	0	R	**Channel 6 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
5	CH5SREQSTATUS	0	R	**Channel 5 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
4	CH4SREQSTATUS	0	R	**Channel 4 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
3	CH3SREQSTATUS	0	R	**Channel 3 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
2	CH2SREQSTATUS	0	R	**Channel 2 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
1	CH1SREQSTATUS	0	R	**Channel 1 Single Request Status**
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			
0	CH0SREQSTATUS	0	R	**Channel 0 Single Request Status**

Bit	Name	Reset	Access	Description
	When this bit is 1, it indicates that the peripheral connected as the input to this DMA channel is requesting the controller to service the DMA channel. The controller services the request by performing the DMA cycle using single DMA transfers.			

H.7.20 DMA_IF - Interrupt Flag Register

Offset	Bit Position																												
0x1000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	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
Name	ERR																				CH11DONE	CH10DONE	CH9DONE	CH8DONE	CH7DONE	CH6DONE	CH5DONE	CH4DONE	CH3DONE	CH2DONE	CH1DONE	CH0DONE

Bit	Name	Reset	Access	Description
31	ERR	0	R	**DMA Error Interrupt Flag**
	This flag is set when an error has occurred on the AHB bus.			
30:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11DONE	0	R	**DMA Channel 11 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
10	CH10DONE	0	R	**DMA Channel 10 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
9	CH9DONE	0	R	**DMA Channel 9 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
8	CH8DONE	0	R	**DMA Channel 8 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
7	CH7DONE	0	R	**DMA Channel 7 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
6	CH6DONE	0	R	**DMA Channel 6 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
5	CH5DONE	0	R	**DMA Channel 5 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
4	CH4DONE	0	R	**DMA Channel 4 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
3	CH3DONE	0	R	**DMA Channel 3 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
2	CH2DONE	0	R	**DMA Channel 2 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
1	CH1DONE	0	R	**DMA Channel 1 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			
0	CH0DONE	0	R	**DMA Channel 0 Complete Interrupt Flag**
	Set when the DMA channel has completed its transfer. If the channel is disabled, the flag is set when there is a request for the channel.			

H.7.21 DMA_IFS - Interrupt Flag Set Register

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

Offset	Bit Position																							
Name	ERR																							
																	CH11DONE	CH10DONE	CH9DONE	CH8DONE	CH7DONE	CH6DONE	CH5DONE	CH4DONE	CH3DONE	CH2DONE	CH1DONE	CH0DONE

Bit	Name	Reset	Access	Description
31	ERR	0	W1	**DMA Error Interrupt Flag Set**
	Set to 1 to set DMA error interrupt flag.			
30:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11DONE	0	W1	**DMA Channel 11 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
10	CH10DONE	0	W1	**DMA Channel 10 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
9	CH9DONE	0	W1	**DMA Channel 9 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
8	CH8DONE	0	W1	**DMA Channel 8 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
7	CH7DONE	0	W1	**DMA Channel 7 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
6	CH6DONE	0	W1	**DMA Channel 6 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
5	CH5DONE	0	W1	**DMA Channel 5 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
4	CH4DONE	0	W1	**DMA Channel 4 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
3	CH3DONE	0	W1	**DMA Channel 3 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
2	CH2DONE	0	W1	**DMA Channel 2 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
1	CH1DONE	0	W1	**DMA Channel 1 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			
0	CH0DONE	0	W1	**DMA Channel 0 Complete Interrupt Flag Set**
	Write to 1 to set the corresponding DMA channel complete interrupt flag.			

H.7.22 DMA_IFC - Interrupt Flag Clear Register

Offset	Bit Position																							
0x1008	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	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
Name	ERR																				CH11DONE	CH10DONE	CH9DONE	CH8DONE	CH7DONE	CH6DONE	CH5DONE	CH4DONE	CH3DONE	CH2DONE	CH1DONE	CH0DONE

Bit	Name	Reset	Access	Description
31	ERR	0	W1	**DMA Error Interrupt Flag Clear**
	Set to 1 to clear DMA error interrupt flag. Note that if an error happened, the Bus Error Clear Register must be used to clear the DMA.			
30:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11DONE	0	W1	**DMA Channel 11 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
10	CH10DONE	0	W1	**DMA Channel 10 Complete Interrupt Flag Clear**

Bit	Name	Reset	Access	Description
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
9	CH9DONE	0	W1	**DMA Channel 9 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
8	CH8DONE	0	W1	**DMA Channel 8 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
7	CH7DONE	0	W1	**DMA Channel 7 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
6	CH6DONE	0	W1	**DMA Channel 6 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
5	CH5DONE	0	W1	**DMA Channel 5 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
4	CH4DONE	0	W1	**DMA Channel 4 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
3	CH3DONE	0	W1	**DMA Channel 3 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
2	CH2DONE	0	W1	**DMA Channel 2 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
1	CH1DONE	0	W1	**DMA Channel 1 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			
0	CH0DONE	0	W1	**DMA Channel 0 Complete Interrupt Flag Clear**
	Write to 1 to clear the corresponding DMA channel complete interrupt flag.			

H.7.23 DMA_IEN - Interrupt Enable register

Offset	Bit Position																											
0x100C	31	30	29	28	27	26	25	24	23	22	21	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	ERR																				CH11DONE	CH10DONE	CH9DONE	CH8DONE	CH7DONE	CH6DONE	CH5DONE	CH4DONE	CH3DONE	CH2DONE	CH1DONE	CH0DONE

Bit	Name	Reset	Access	Description
31	ERR	0	RW	**DMA Error Interrupt Flag Enable**
	Set this bit to enable interrupt on AHB bus error.			
30:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	CH11DONE	0	RW	**DMA Channel 11 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
10	CH10DONE	0	RW	**DMA Channel 10 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
9	CH9DONE	0	RW	**DMA Channel 9 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
8	CH8DONE	0	RW	**DMA Channel 8 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
7	CH7DONE	0	RW	**DMA Channel 7 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
6	CH6DONE	0	RW	**DMA Channel 6 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
5	CH5DONE	0	RW	**DMA Channel 5 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
4	CH4DONE	0	RW	**DMA Channel 4 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			

Bit	Name	Reset	Access	Description
3	CH3DONE	0	RW	**DMA Channel 3 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
2	CH2DONE	0	RW	**DMA Channel 2 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
1	CH1DONE	0	RW	**DMA Channel 1 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			
0	CH0DONE	0	RW	**DMA Channel 0 Complete Interrupt Enable**
	Write to 1 to enable complete interrupt on this DMA channel. Clear to disable the interrupt.			

H.7.24 DMA_CTRL - DMA Control Register

Offset	Bit Position																													
0x1010	31	30	29	28	27	26	25	24	23	22	21	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																															PRDU	DESCRECT

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	PRDU	0	RW	**Prevent Rect Descriptor Update**
	Allows the reuse of a rect descriptor. When active CH0 and no others can have RDS set			
0	DESCRECT	0	RW	**Descriptor Specifies Rectangle**
	Word 4 in dma descriptor specifies WIDTH, HEIGHT and SRCSTRIDE for rectangle copies. WIDTH is given by bits 9:0, HEIGHT is given by bits 19:10, and SRCSTRIDE is given by bits 30:20			

H.7.25 DMA_RDS - DMA Retain Descriptor State

Offset	Bit Position																													
0x1014	31	30	29	28	27	26	25	24	23	22	21	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	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																					RDSCH11	RDSCH10	RDSCH9	RDSCH8	RDSCH7	RDSCH6	RDSCH5	RDSCH4	RDSCH3	RDSCH2	RDSCH1	RDSCH0

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	RDSCH11	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
10	RDSCH10	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
9	RDSCH9	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
8	RDSCH8	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			

Bit	Name	Reset	Access	Description
7	RDSCH7	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
6	RDSCH6	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
5	RDSCH5	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
4	RDSCH4	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
3	RDSCH3	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
2	RDSCH2	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
1	RDSCH1	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			
0	RDSCH0	0	RW	**Retain Descriptor State**
	Speed up execution of consecutive DMA requests from the same channel by not reading descriptor at the start of every arbitration cycle if the next channel is the same as the previous			

H.7.26 DMA_LOOP0 - Channel 0 Loop Register

Offset	Bit Position																														
0x1020	31	30	29	28	27	26	25	24	23	22	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							RWH									
Name																EN							WIDTH									

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	EN	0	RW	**DMA Channel 0 Loop Enable**
	Loop enable for channel 0			
15:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9:0	WIDTH	0x000	RWH	**Loop Width**
	Reload value for N_MINUS_1 when loop is enabled			

H.7.27 DMA_LOOP1 - Channel 1 Loop Register

Offset	Bit Position																														
0x1024	31	30	29	28	27	26	25	24	23	22	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																EN							WIDTH									

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	EN	0	RW	**DMA Channel 1 Loop Enable**
	Loop enable for channel 1			
15:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9:0	WIDTH	0x000	RW	**DMA Channel 1 Loop Width**
	Reload value for N_MINUS_1 when loop is enabled			

H.7.28 DMA_RECT0 - Channel 0 Rectangle Register

Offset	Bit Position																													
0x1060	31	30	29	28	27	26	25	24	23	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											0x000								
Access	RW											RWH											RWH								
Name	DSTSTRIDE											SRCSTRIDE											HEIGHT								

Bit	Name	Reset	Access	Description
31:21	DSTSTRIDE	0x000	RW	**DMA Channel 0 Destination Stride**
	Space between start of lines in destination rectangle			
20:10	SRCSTRIDE	0x000	RWH	**DMA Channel 0 Source Stride**
	Space between start of lines in source rectangle			
9:0	HEIGHT	0x000	RWH	**DMA Channel 0 Rectangle Height**
	Number of lines when doing rectangle copy. Set to the number of lines - 1.			

H.7.29 DMA_CHx_CTRL - Channel Control Register

Offset	Bit Position																													
0x1100	31	30	29	28	27	26	25	24	23	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.		
21:16	SOURCESEL	0x00	RW	**Source Select**
	Select input source to DMA channel.			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0b000000	NONE	No source selected
0b001000	ADC0	Analog to Digital Converter 0
0b001010	DAC0	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
0b010000	LEUART0	Low Energy UART 0
0b010001	LEUART1	Low Energy UART 1
0b010100	I2C0	I2C 0
0b010101	I2C1	I2C 1
0b011000	TIMER0	Timer 0
0b011001	TIMER1	Timer 1
0b011010	TIMER2	Timer 2
0b011011	TIMER3	Timer 3
0b101100	UART0	Universal Asynchronous Receiver/Transmitter 0
0b101101	UART1	Universal Asynchronous Receiver/Transmitter 1
0b110000	MSC	
0b110001	AES	Advanced Encryption Standard Accelerator
0b110010	LESENSE	Low Energy Sensor Interface

15:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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 = 0b001000 (ADC0)		
0b0000	ADC0SINGLE	ADC0SINGLE
0b0001	ADC0SCAN	ADC0SCAN
SOURCESEL = 0b001010 (DAC0)		
0b0000	DAC0CH0	DAC0CH0
0b0001	DAC0CH1	DAC0CH1
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	USART1RXDATAVRIGHT	USART1RXDATAVRIGHT REQ/SREQ
0b0100	USART1TXBLRIGHT	USART1TXBLRIGHT REQ/SREQ
SOURCESEL = 0b001110 (USART2)		
0b0000	USART2RXDATAV	USART2RXDATAV REQ/SREQ
0b0001	USART2TXBL	USART2TXBL REQ/SREQ
0b0010	USART2TXEMPTY	USART2TXEMPTY
0b0011	USART2RXDATAVRIGHT	USART2RXDATAVRIGHT REQ/SREQ
0b0100	USART2TXBLRIGHT	USART2TXBLRIGHT REQ/SREQ

Bit	Name	Reset	Access	Description
3:0	SIGSEL	0x0	RW	**Signal Select**

Select input signal to DMA channel.

Bit	Name	Reset	Access	Description

Value	Mode	Description
SOURCESEL = 0b000000 (NONE)		
0bxxxx	OFF	Channel input selection is turned off
SOURCESEL = 0b001000 (ADC0)		
0b0000	ADC0SINGLE	ADC0SINGLE
0b0001	ADC0SCAN	ADC0SCAN
SOURCESEL = 0b010000 (LEUART0)		
0b0000	LEUART0RXDATAV	LEUART0RXDATAV
0b0001	LEUART0TXBL	LEUART0TXBL
0b0010	LEUART0TXEMPTY	LEUART0TXEMPTY
SOURCESEL = 0b010001 (LEUART1)		
0b0000	LEUART1RXDATAV	LEUART1RXDATAV
0b0001	LEUART1TXBL	LEUART1TXBL
0b0010	LEUART1TXEMPTY	LEUART1TXEMPTY
SOURCESEL = 0b010100 (I2C0)		
0b0000	I2C0RXDATAV	I2C0RXDATAV
0b0001	I2C0TXBL	I2C0TXBL
SOURCESEL = 0b010101 (I2C1)		
0b0000	I2C1RXDATAV	I2C1RXDATAV
0b0001	I2C1TXBL	I2C1TXBL
SOURCESEL = 0b011000 (TIMER0)		
0b0000	TIMER0UFOF	TIMER0UFOF
0b0001	TIMER0CC0	TIMER0CC0
0b0010	TIMER0CC1	TIMER0CC1
0b0011	TIMER0CC2	TIMER0CC2
SOURCESEL = 0b011001 (TIMER1)		
0b0000	TIMER1UFOF	TIMER1UFOF
0b0001	TIMER1CC0	TIMER1CC0
0b0010	TIMER1CC1	TIMER1CC1
0b0011	TIMER1CC2	TIMER1CC2
SOURCESEL = 0b011010 (TIMER2)		
0b0000	TIMER2UFOF	TIMER2UFOF
0b0001	TIMER2CC0	TIMER2CC0
0b0010	TIMER2CC1	TIMER2CC1
0b0011	TIMER2CC2	TIMER2CC2
SOURCESEL = 0b011011 (TIMER3)		
0b0000	TIMER3UFOF	TIMER3UFOF
0b0001	TIMER3CC0	TIMER3CC0
0b0010	TIMER3CC1	TIMER3CC1
0b0011	TIMER3CC2	TIMER3CC2
SOURCESEL = 0b101100 (UART0)		
0b0000	UART0RXDATAV	UART0RXDATAV REQ/SREQ
0b0001	UART0TXBL	UART0TXBL REQ/SREQ
0b0010	UART0TXEMPTY	UART0TXEMPTY

Bit	Name	Reset	Access	Description
3:0	SIGSEL	0x0	RW	**Signal Select**

Select input signal to DMA channel.

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	SOURCESEL = 0b101101 (UART1)			
	0b0000	UART1RXDATAV		UART1RXDATAV REQ/SREQ
	0b0001	UART1TXBL		UART1TXBL REQ/SREQ
	0b0010	UART1TXEMPTY		UART1TXEMPTY
	SOURCESEL = 0b110000 (MSC)			
	0b0000	MSCWDATA		MSCWDATA
	SOURCESEL = 0b110001 (AES)			
	0b0000	AESDATAWR		AESDATAWR
	0b0001	AESXORDATAWR		AESXORDATAWR
	0b0010	AESDATARD		AESDATARD
	0b0011	AESKEYWR		AESKEYWR
	SOURCESEL = 0b110010 (LESENSE)			
	0b0000	LESENSEBUFDATAV		LESENSEBUFDATAV REQ/SREQ

I ARM Memory System Controller

I.1 Introduction

The Memory System Controller (MSC) is the program memory unit of the ARM core. The flash memory is readable and writable from both the Cortex-M3 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. Read and write operations are supported in the energy modes EM0 and EM1.

I.2 Features

- AHB read interface
  - Scalable access performance to optimize the Cortex-M3 code interface
    - Zero wait-state access up to 16 MHz and one wait-state for up to 32 MHz and two wait-states for up to 48 MHz
    - Advanced energy optimization functionality
      - Conditional branch target prefetch suppression
      - Cortex-M3 disfolding of if-then (IT) blocks
      - Instruction Cache
      - Instruction Prefetch
  - DMA read support in EM0 and EM1
- Command and status interface
  - Flash write and erase
    - Accessible from Cortex-M3 in EM0
    - DMA write support in EM0 and EM1
    - Read-while-write support
    - Write two words at a time
  - 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
  - User data lock bits
- End-of-write and end-of-erase interrupts

I.3 Functional Description

The size of the main block is device dependent. The largest size available is 1024 kB (256 pages). The information block has 4096 bytes 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 64 outlines how the Flash is mapped in the memory space. All Flash memory is organized into 4096 byte pages.

Block	Page	Base address	Write/Erase by	Software readable	Purpose/Name	Size
Main[7]	0	0x00000000	Software, debug	Yes	User code and data	512 kB - 1024 kB
	.		Software, debug	Yes		
	255	0x000FF000	Software, debug	Yes		
Reserved	-	0x00100000	-	-	Reserved for flash expansion	~24 MB
Information	0	0x0FE00000	Software, debug	Yes	User Data (UD)	4 kB
	-	0x0FE00800	-	-	Reserved	
	1	0x0FE04000	Write: Software, debug Erase: Debug only	Yes	Lock Bits (LB)	4 kB
	-	0x0FE04800	-	-	Reserved	
	2	0x0FE08000	-	Yes	Device Information (DI)	4 kB
	-	0x0FE08800	-	-	Reserved	
Reserved	-	0x0FE10000	-	-	Reserved for flash expansion	Rest of code space

Figure 64: MSC Flash Memory Mapping

I.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 [?].

I.3.2 Lock Bits (LB) Page Description

This page contains the following information:

- Debug Lock Word (DLW)
- User data page Lock Word (ULW)
- Mass erase Lock Word (MLW)
- Main block Page Lock Words (PLWs)

The words in this page are organized as shown in Table 65:

[7] Block/page erased by a device erase

127	DLW
126	ULW
125	MLW
N	PLW[N]
...	...
1	PLW[1]
0	PLW[0]

Figure 65: Lock Bits Page Structure

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. If the bits are not 0xF, then debug access to the core is locked. See [?] for details on how to unlock the debug access.

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.

Word 125 is the mass erase lock word (MLW). Bit 0 locks the lower half of the flash, preventing mass erase, and bit 1 locks the upper half of the flash. The mass erase lock bits will not have any effect on device erases initiated from the Authentication Access Port (AAP) registers. The AAP is described in more detail in [?].

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. Similarly, PLW[1] contains lock bits for page 32-63 and so on. A page is locked when the bit is 0. A locked page cannot be erased or written.

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 [?]. Note that the AAP is only accessible from the debug interface, and cannot be accessed from the Cortex-M3 core.

I.3.3 Device Information (DI) Page

This read-only page holds the calibration data for the oscillator and other analog peripherals from the production test as well as a unique device ID. The page is further described in [?].

I.3.4 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.

One Wait-state Access

After reset, the HFCORECLK is normally 14 MHz from the HFRCO and the MODE field of the MSC_READCTRL register is set to WS1 (one wait-state). The reset value must be WS1 as an uncalibrated HFRCO may produce a frequency higher than 16 MHz. Software must not select a zero wait-state mode unless the clock is guaranteed to be 16 MHz or below, otherwise the resulting behavior is undefined. If a HFCORECLK frequency above 16 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 can be set to WS0 or WS0SCBTP, but only after the frequency transition is completed. If the HFRCO is used, wait until the oscillator is stable on the new frequency. Otherwise, the behavior is unpredictable.

To run at a frequency higher than 32 MHz, WS2 or WS2SCBTP must be selected to insert two wait-states for every flash access.

Zero Wait-state Access

At 16 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 from 16 MHz and below. By default, the Cortex-M3 uses speculative prefetching and If-Then block folding to maximize code execution performance at the cost of additional flash accesses and energy consumption.

Operation Above 32 MHz

To run at frequencies higher than 32 MHz, MODE in MSC_READCTRL must be set to WS2 or WS2SCBTP.

Suppressed Conditional Branch Target Prefetch (SCBTP)

MSC offers a special instruction fetch mode which optimizes energy consumption by cancelling Cortex-M3 conditional branch target prefetches. Normally, the Cortex-M3 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 16 MHz and below. To enable the mode at frequencies from 16 MHz and below write WS0SCBTP to the MODE field of the MSC_READCTRL register. For frequencies above 16 MHz, use the WS1SCBTP mode, and for frequencies above 32 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.

Cortex-M3 If-Then Block Folding

The Cortex-M3 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-M3 Technical Reference Manual for details. Normally, it is expected that this feature is most efficient at core frequencies above 16 MHz. Folding is enabled by default.

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 16 MHz).

The instruction cache is connected directly to the Cortex-M3 and functions as a memory access filter between the processor and the memory system, as illustrated in Figure 66. The cache consists of an access filter, lookup logic, a 128x32 SRAM (512 bytes) and two performance counters. The access filter checks that the address for the access is of an instruction in the code space (instructions in RAM outside the code space 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 cache consists of 16 8-word cachelines organized as 4 sets with 4 ways. 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.

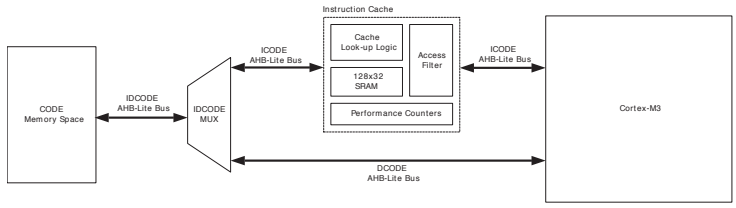


Figure 66: Instruction Cache

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.

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 outside the code space. 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 execute 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.

Instruction Prefetch

The MSC also includes instruction prefetch capability for the internal flash memory. This feature is by default disabled, but can be enabled by setting PREFETCH in MSC_READCTRL. The prefetcher works by the assumption that the next instruction word will be needed in the next fetch. This next word is fetched before the word is actually needed. If it turns out that this next word is actually needed by the CPU, the prefetched word can be returned in one clock cycle, removing the wait-states that would otherwise potentially be needed by a flash read. With the prefetcher enabled, the number of waitstates to the flash when accessing code that is not in the cache is effectively halved.

I.3.5 Erase and Write Operations

The AUXHFRCO is used for timing during flash write and erase operations. To achieve correct timing, the MSC_TIMEBASE register has to be configured according to the settings in CMU_AUXHFRCOCTRL. BASE in MSC_TIMEBASE defines how many AUXCLK cycles - 1 there is in 1 us or 5 us, depending on the configuration of PERIOD. To ensure that timing of flash write and erase operations is within the specification of the flash, the value written to BASE should give at least a 10% margin with respect to the period, i.e. for the 1 us PERIOD, the number of cycles should at least span 1.1 us, and for the 5 us period they should span at least 5.5 us. For the 1 MHz band, PERIOD in MSC_TIMEBASE should be set to 5US, while it should be set to 1US for all other AUXHFRCO bands.

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)

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.

The MSC_WDATA and MSC_ADDRB registers are not retained when entering EM2 or lower energy modes.

Double-Word Writes The ARM core can do double writes to the flash. This is enabled by setting WDOUBLE in MSC_WRITECTRL, and only has effect on the main pages of the flash. When double writes are enabled, MSC_WDATA accepts two words before a flash write is started. This is signaled by WDATAREADY in MSC_STATUS going high again after the first word is written. When the second word is written, the actual write begins. **Low-Power Erase** Because of the fast flash in ARM core these devices consume approximately twice the amount of current while doing erase operations when compared to other EFM32 devices. To limit the maximum erase current, the erase operation can be slowed down. Set LPERASE in MSC_WRITECTRL to double the erase time, halving the erase current. **Low-Power Write** The maximum write current can also be limited, by slowing down write operations by setting LPWRITE in MSC_WRITECTRL. For single-word writes, this has no effect on write time, but for consecutive writes, the write-time doubles. LPWRITE cannot be set while WDOUBLE is set. **Read-While-Write**

Reading from the lower half of the flash is possible while writing/erasing from the upper half of the flash and vice versa. Enable read-while-write by setting RWWEN in MSC_WRITECTRL.

The information pages can be written and erased while running code from the lower half of the flash.

Mass erase

A mass erase can be initiated from software using ERASEMAIN0 and ERASEMAIN1 in MSC_WRITECMD. These commands will start a masserase on the lower and upper half of the flash respectively. 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).

I.4 Register Map

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

Offset	Name	Type	Description
0x000	MSC_CTRL	RW	Memory System Control Register
0x004	MSC_READCTRL	RW	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
0x02C	MSC_IF	R	Interrupt Flag Register
0x030	MSC_IFS	W1	Interrupt Flag Set Register
0x034	MSC_IFC	W1	Interrupt Flag Clear Register
0x038	MSC_IEN	RW	Interrupt Enable Register
0x03C	MSC_LOCK	RW	Configuration Lock Register
0x040	MSC_CMD	W1	Command Register
0x044	MSC_CACHEHITS	R	Cache Hits Performance Counter
0x048	MSC_CACHEMISSES	R	Cache Misses Performance Counter
0x050	MSC_TIMEBASE	RW	Flash Write and Erase Timebase
0x054	MSC_MASSLOCK	RW	Mass Erase Lock Register

I.5 Register Description

I.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																																1
Access																																RW
Name																																BUSFAULT

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	BUSFAULT	1	RW	**Bus Fault Response Enable**
	When this bit is set, the memory system generates bus error response.			

Value	Mode	Description
0	GENERATE	A bus fault is generated on access to unmapped code and system space.
1	IGNORE	Accesses to unmapped address space is ignored.

I.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															0x0									0	0	0	0	0	0	0x1		
Access															RW									RW	RW	RW	RW	RW	RW	RW		
Name															BUSSTRATEGY									PREFETCH	RAMCEN	EBICDIS	ICCDIS	AIDIS	IFCDIS	MODE		

Bit	Name	Reset	Access	Description
31:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17:16	BUSSTRATEGY	0x0	RW	**Strategy for bus matrix**
	Specify which master has low latency to bus matrix.			

Value	Mode	Description
0	CPU	
1	DMA	
2	DMAEM1	
3	NONE	

15:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8	PREFETCH	0	RW	**Prefetch Mode**
	Set to configure level of prefetching.			
7	RAMCEN	0	RW	**RAM Cache Enable**
	Enable instruction caching for RAM in code-space.			
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.			

Bit	Name	Reset	Access	Description
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	MODE	0x1	RW	**Read Mode**

If software wants to set a core clock frequency above 16 MHz, this register must be set to WS1 or WS1SCBTP before the core clock is switched to the higher frequency. When changing to a lower frequency, this register can be set to WS0 or WS0SCBTP after the frequency transition has been completed. After reset, the core clock is 14 MHz from the HFRCO but the MODE field of MSC_READCTRL register is set to WS1. This is because the HFRCO may produce a frequency above 16 MHz before it is calibrated. If the HFRCO is used as clock source, wait until the oscillator is stable on the new frequency to avoid unpredictable behavior.

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. This mode is required for a core frequency above 16 MHz.
2	WS0SCBTP	Zero wait-states inserted with the Suppressed Conditional Branch Target Prefetch (SCBTP) function enabled. SCBTP saves energy by delaying the Cortex' 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.
3	WS1SCBTP	One wait-state access with SCBTP enabled.
4	WS2	Two wait-states inserted for each fetch or read transfer. This mode is required for a core frequency above 32 MHz.
5	WS2SCBTP	Two wait-state access with SCBTP enabled.

I.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	0	0	0
Access																											RW	RW	RW	RW	RW	RW
Name																											RWWEN	LPERASE	LPWRITE	WDOUBLE	IRQERASEABORT	WREN

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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	LPERASE	0	RW	**Low-Power Erase**
	When set, the erase time doubles while halving the erase current.			
3	LPWRITE	0	RW	**Low-Power Erase**
	When set, write times might double while reducing current consumption.			
2	WDOUBLE	0	RW	**Write two words at a time**
	When set, two words are written to the flash at a time.			
1	IRQERASEABORT	0	RW	**Abort Page Erase on Interrupt**
	When this bit is set to 1, any Cortex 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.			

I.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.		
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.		
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.		
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**
	Functions like MSC_CMD_WRITEONCE, but will set MSC_STATUS_WORDTIMEOUT if no new data is written to MSC_WDATA within the 30 μs timeout.			
3	WRITEONCE	0	W1	**Word Write-Once 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 30 μs 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.			
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.			
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.			

I.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**

Bit	Name	Reset	Access	Description
	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. The MSC_ADDR register is not readable. This register is not retained when entering EM2 or lower energy modes.			

I.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, otherwise the data is ignored. This register is not retained when entering EM2 or lower energy modes.			

I.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																										0	0	0	1	0	0	0
Access																										R	R	R	R	R	R	R
Name																										PCRUNNING	ERASEABORTED	WORDTIMEOUT	WDATAREADY	INVADDR	LOCKED	BUSY

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

I.5.8 MSC_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
Access																													R	R	R	R
Name																													CMOF	CHOF	WRITE	ERASE

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	CMOF	0	R	**Cache Misses Overflow Interrupt Flag**
	Set when MSC_CACHEMISSES overflows.			
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.			

I.5.9 MSC_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
Access																													W1	W1	W1	W1
Name																													CMOF	CHOF	WRITE	ERASE

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	CMOF	0	W1	**Cache Misses Overflow Interrupt Set**
	Set the CMOF flag and generate interrupt.			
2	CHOF	0	W1	**Cache Hits Overflow Interrupt Set**
	Set the CHOF flag and generate interrupt.			
1	WRITE	0	W1	**Write Done Interrupt Set**
	Set the write done bit and generate interrupt.			
0	ERASE	0	W1	**Erase Done Interrupt Set**
	Set the erase done bit and generate interrupt.			

I.5.10 MSC_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
Access																													W1	W1	W1	W1
Name																													CMOF	CHOF	WRITE	ERASE

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	CMOF	0	W1	**Cache Misses Overflow Interrupt Clear**
	Clear the CMOF interrupt flag.			
2	CHOF	0	W1	**Cache Hits Overflow Interrupt Clear**
	Clear the CHOF interrupt flag.			
1	WRITE	0	W1	**Write Done Interrupt Clear**
	Clear the write done bit.			
0	ERASE	0	W1	**Erase Done Interrupt Clear**
	Clear the erase done bit.			

I.5.11 MSC_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
Access																													RW	RW	RW	RW
Name																													CMOF	CHOF	WRITE	ERASE

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	CMOF	0	RW	**Cache Misses Overflow Interrupt Enable**
	Enable the cache misses performance counter overflow interrupt.			
2	CHOF	0	RW	**Cache Hits Overflow Interrupt Enable**
	Enable the cache hits performance counter overflow interrupt.			
1	WRITE	0	RW	**Write Done Interrupt Enable**
	Enable the write done interrupt.			
0	ERASE	0	RW	**Erase Done Interrupt Enable**
	Enable the erase done interrupt.			

I.5.12 MSC_LOCK - Configuration Lock Register

Offset	Bit Position																														
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0000															
Access																	RW															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	LOCKKEY	0x0000	RW	**Configuration Lock**
	Write any other value than the unlock code to lock access to MSC_CTRL, MSC_READCTRL, MSC_WRITECTRL and MSC_TIMEBASE. Write the unlock code to enable access. When reading the register, bit 0 is set when the lock is enabled.			

Bit	Name	Reset	Access	Description

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.

I.5.13 MSC_CMD - Command 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
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.		
2	STOPPC	0	W1	**Stop Performance Counters**
	Use this command 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.			

I.5.14 MSC_CACHEHITS - Cache Hits Performance Counter

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													0x00000																			
Access													R																			
Name													CACHEHITS																			

Bit	Name	Reset	Access	Description
31:20	Reserved	To ensure compatibility with future devices, always write bits to 0.		
19:0	CACHEHITS	0x00000	R	**Cache hits since last performance counter start command.**
	Use to measure cache performance for a particular code section.			

I.5.15 MSC_CACHEMISSES - Cache Misses 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																			

Offset	Bit Position	
Access		R
Name		CACHEMISSES

Bit	Name	Reset	Access	Description
31:20	Reserved	To ensure compatibility with future devices, always write bits to 0.		
19:0	CACHEMISSES	0x00000	R	**Cache misses since last performance counter start command.**
	Use to measure cache performance for a particular code section.			

I.5.16 MSC_TIMEBASE - Flash Write and Erase Timebase

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											0x10					
Access																RW											RW					
Name																PERIOD											BASE					

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	PERIOD	0	RW	**Sets the timebase period**
	Decides whether TIMEBASE specifies the number of AUX cycles in 1 us or 5 us. 5 us should only be used with 1 MHz AUXHFRCO band.			

Value	Mode	Description
0	1US	TIMEBASE period is 1 us.
1	5US	TIMEBASE period is 5 us.

Bit	Name	Reset	Access	Description
15:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:0	BASE	0x10	RW	**Timebase used by MSC to time flash writes and erases**
	Should be set to the number of full AUX clock cycles in the period given by MSC_TIMEBASE_PERIOD. I.e. 1.1 us or 5.5. us with PERIOD cleared or set, respectively. The resetvalue of the timebase matches a 14 MHz AUXHFRCO, which is the default frequency of the AUXHFRCO.			

I.5.17 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																	RW															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	LOCKKEY	0x0001	RW	**Mass Erase Lock**
	Write any other value than the unlock code to lock access the the ERASEMAIN0 and ERASEMAIN1 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.			

Bit	Name	Reset	Access	Description
	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.

J ARM Reset Management Unit

J.1 Introduction

The RMU is responsible for handling the reset functionality of the ARM core.

J.2 Features

- Reset sources
  - Power-on Reset (POR)
  - Brown-out Detection (BOD) on the following power domains:
    - Regulated domain
    - Unregulated domain
    - Analog Power Domain 0 (AVDD0)
    - Analog Power Domain 1 (AVDD1)
  - RESETn pin reset
  - Watchdog reset
  - EM4 wakeup reset from pin
  - EM4 wakeup reset from Backup RTC interrupt
  - Wakeup from Backup Mode
  - Software triggered reset (SYSRESETREQ)
  - Core LOCKUP condition
- EM4 Detection
- A software readable register indicates the cause of the last reset

J.3 Functional Description

The RMU monitors each of the reset sources of the ARM core. If one or more reset sources go active, the RMU applies reset to the ARM core. When the reset sources go inactive the ARM core starts up. At startup the ARM core loads the stack pointer and program entry point from memory, and starts execution.

As seen in Figure 67 the Power-on Reset, Brown-out Detectors, Watchdog timeout and RESETn pin all reset the whole system including the Debug Interface. A Core Lockup condition or a System reset request from software resets the whole system except the Debug Interface.

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 must be cleared by software.

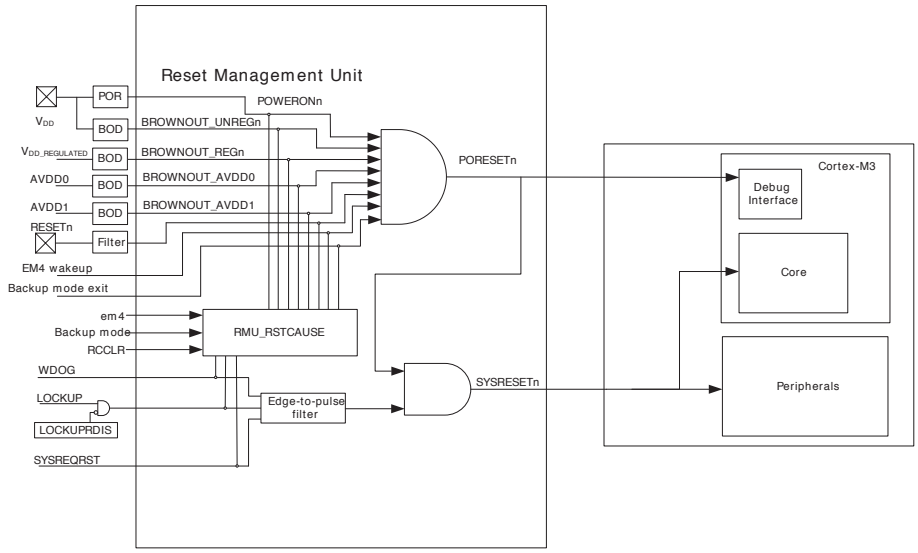


Figure 67: RMU Reset Input Sources and Connections.

J.3.1 RMU_RSTCAUSE Register

The RMU_RSTCAUSE register indicates the reason for the last reset. 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.

The following procedure must be done to clear RMU_RSTCAUSE:

1. Write a 1 to RCCLR in RMU_CMD
2. Write a 1 to bit 0 in EMU_AUXCTRL
3. Write a 0 to bit 0 in EMU_AUXCTRL

RMU_RSTCAUSE should be interpreted according to Table 68. X bits are don't care. Notice that it is possible to have multiple reset causes. For example, an external reset and a watchdog reset may happen simultaneously.

When exiting EM4 with external reset, both the BODREGRST and BODUNREGRST in RSTCAUSE might be set (i.e. are invalid)

J.3.2 Power-On Reset (POR)

The POR ensures that the ARM core does not start up before the supply voltage V_{DD} has reached the threshold voltage VPORthr (see Device Datasheet Electrical Characteristics for details). Before the threshold voltage is reached, the ARM core is kept in reset state. The operation of the POR is illustrated in Figure 69, with the

Register Value	Cause
0bXXXX XXXX XXXX XXX1	A Power-on Reset has been performed. X bits are don't care.
0bXXXX XXXX 0XXX XX10	A Brown-out has been detected on the unregulated power.
0bXXXX XXXX XXX0 0100	A Brown-out has been detected on the regulated power.
0bXXXX XXXX XXXX 1X00	An external reset has been applied.
0bXXXX XXXX XXX1 XX00	A watchdog reset has occurred.
0bXXXX X000 0010 0000	A lockup reset has occurred.
0bXXXX X000 01X0 0000	A system request reset has occurred.
0bXXXX X000 1XX0 0XX0	The system has woken up from EM4.
0bXXXX X001 1XX0 0XX0	The system has woken up from EM4 on an EM4 wakeup reset request from pin.
0bXXXX X01X XXX0 0000	A Brown-out has been detected on Analog Power Domain 0 (AVDD0).
0bXXXX X10X XXX0 0000	A Brown-out has been detected on Analog Power Domain 1 (AVDD1).
0bXXXX 1XXX XXXX 0XX0	A Brown-out has been detected by the Backup BOD on VDD_DREG.
0bXXX1 XXXX XXXX 0XX0	A Brown-out has been detected by the Backup BOD on BU_VIN.
0bXX1X XXXX XXXX 0XX0	A Brown-out has been detected by the Backup BOD on unregulated power
0bX1XX XXXX XXXX 0XX0	A Brown-out has been detected by the Backup BOD on regulated power.
0b1XXX XXXX XXXX XXX0	The system has been in Backup mode.

Figure 68: RMU Reset Cause Register Interpretation

active low POWERONn reset signal. The reason for the "unknown" region is that the corresponding supply voltage is too low for any reliable operation.

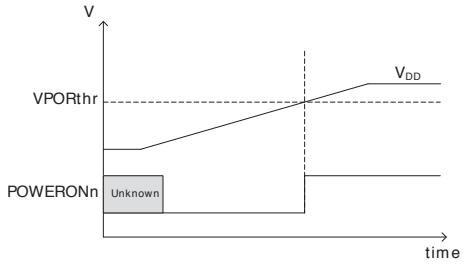


Figure 69: RMU Power-on Reset Operation

J.3.3 Brown-Out Detector Reset (BOD)

The ARM core has 4 brownout detectors, one for the unregulated 3.0 V power, one for the regulated internal power, one for Analog Power Domain 0 (AVDD0), and one for Analog Power Domain 1 (AVDD1). The BODs are constantly monitoring the voltages. Whenever the unregulated or regulated power drops below the VBODthr value (see Electrical Characteristics for details), or if the AVDD0 or AVDD1 drops below the voltage at the decouple pin (DEC), 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 bods drops below decouple pin (DEC). The operation of the BOD is illustrated in Figure 70. The "unknown" regions are handled by the POR module.

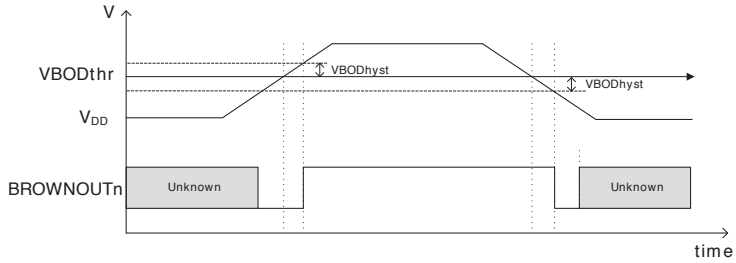


Figure 70: RMU Brown-out Detector Operation

J.3.4 RESETn pin Reset

Forcing the RESETn pin low generates a reset of the ARM core. 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 ARM core.

J.3.5 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.

J.3.6 Lockup Reset

A Cortex-M3 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.

For more information about the Cortex-M3 lockup conditions see the ARMv7-M Architecture Reference Manual. The Lockup reset does not reset the Debug

Interface. Set the LOCKUPRDIS bit in the RMU_CTRL register in order to disable this reset source.

J.3.7 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 (write 0x05FA 0004), a reset is issued. The SYSRESETREQ does not reset the Debug Interface.

J.3.8 EM4 Reset

Whenever EM4 is entered, the EM4RST bit is set. This bit enables the user to identify that the device has been in EM4. Upon wake-up this bit should be cleared by software.

J.3.9 EM4 Wakeup Reset

Whenever the system is woken up from EM4 on a pin wake-up request, the EM4WURST bit is set. This bit enables the user to identify that the device was woken up from EM4 using a pin wake-up request. Upon wake-up this bit should be cleared by software.

J.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

J.5 Register Description

J.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																															1	0
Access																															RW	RW
Name																															BURSTEN	LOCKUPRDIS

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	BURSTEN	1	RW	**Backup domain reset enable**
	This bit has to be cleared before accessing the registers in the BURTC.			
0	LOCKUPRDIS	0	RW	**Lockup Reset Disable**
	Set this bit to disable the LOCKUP signal (from the Cortex) from resetting the device.			

J.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	0	0	0	0	0	0
Access																	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name																	BUMODERST	BUBODREG	BUBODUNREG	BUBODBUVIN	BUBODVDDDREG	BODAVDD1	BODAVDD0	EM4WURST	EM4RST	SYSREQRST	LOCKUPRST	WDOGRST	EXTRST	BODREGRST	BODUNREGRST	PORST

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15	BUMODERST	0	R	**Backup mode reset**
	Set if the system has been in Backup mode. Must be cleared by software.			

Bit	Name	Reset	Access	Description
14	BUBODREG	0	R	**Backup Brown Out Detector Regulated Domain**
	Set if the Backup BOD sensing on regulated power triggers. Must be cleared by software.			
13	BUBODUNREG	0	R	**Backup Brown Out Detector Unregulated Domain**
	Set if the Backup BOD sensing on unregulated power triggers. Must be cleared by software.			
12	BUBODBUVIN	0	R	**Backup Brown Out Detector, BU_VIN**
	Set if the Backup BOD sensing on BU_VIN triggers. Must be cleared by software.			
11	BUBODVDDDREG	0	R	**Backup Brown Out Detector, VDD_DREG**
	Set if the Backup BOD sensing on VDDD_REG triggers. Must be cleared by software.			
10	BODAVDD1	0	R	**AVDD1 Bod Reset**
	Set if analog power domain 1 brown out detector reset has been performed. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
9	BODAVDD0	0	R	**AVDD0 Bod Reset**
	Set if analog power domain 0 brown out detector reset has been performed. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
8	EM4WURST	0	R	**EM4 Wake-up Reset**
	Set if the system has been woken up from EM4 from a reset request from pin. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
7	EM4RST	0	R	**EM4 Reset**
	Set if the system has been in EM4. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
6	SYSREQRST	0	R	**System Request Reset**
	Set if a system request reset has been performed. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
5	LOCKUPRST	0	R	**LOCKUP Reset**
	Set if a LOCKUP reset has been requested. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
4	WDOGRST	0	R	**Watchdog Reset**
	Set if a watchdog reset has been performed. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
3	EXTRST	0	R	**External Pin Reset**
	Set if an external pin reset has been performed. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
2	BODREGRST	0	R	**Brown Out Detector Regulated Domain Reset**
	Set if a regulated domain brown out detector reset has been performed. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
1	BODUNREGRST	0	R	**Brown Out Detector Unregulated Domain Reset**
	Set if a unregulated domain brown out detector reset has been performed. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			
0	PORST	0	R	**Power On Reset**
	Set if a power on reset has been performed. Must be cleared by software. Please see Table 68 for details on how to interpret this bit.			

J.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.		
0	RCCLR	0	W1	**Reset Cause Clear**
	Set this bit to clear the LOCKUPRST and SYSREQRST bits in the RMU_RSTCAUSE register. Use the HRCCLR bit in the EMU_AUXCTRL register to clear the remaining bits.			

K ARM Energy Management Unit

K.1 Introduction

The Energy Management Unit (EMU) manages all the low energy modes (EM) in the ARM core. Each energy mode manages if the CPU and the various peripherals are available. The energy modes range from EM0 to EM4, where EM0, also called run mode, enables the CPU and all peripherals. The lowest recoverable energy mode, EM3, disables the CPU and most peripherals while maintaining wake-up and RAM functionality. EM4 disables everything except the POR, pin reset and optionally Backup RTC, 512 byte data retention, GPIO state retention, and EM4 reset wakeup request.

The various energy modes differ in:

- Energy consumption
- CPU activity
- Reaction time
- Wake-up triggers
- Active peripherals
- Available clock sources

Low energy modes EM1 to EM4 are enabled through the application software. In EM1-EM3, a range of wake-up triggers return the microcontroller back to EM0. EM4 can only return to EM0 by power on reset, external pin reset, EM4 GPIO wakeup request, or Backup RTC interrupt.

The EMU can also be used to turn off the power to unused SRAM blocks.

K.2 Features

- Energy Mode control from software
- Flexible wakeup from low energy modes
- Low wakeup time

K.3 Functional Description

The Energy Management Unit (EMU) is responsible for managing the wide range of energy modes available in the ARM core. An overview of the EMU module is shown in Figure 71.

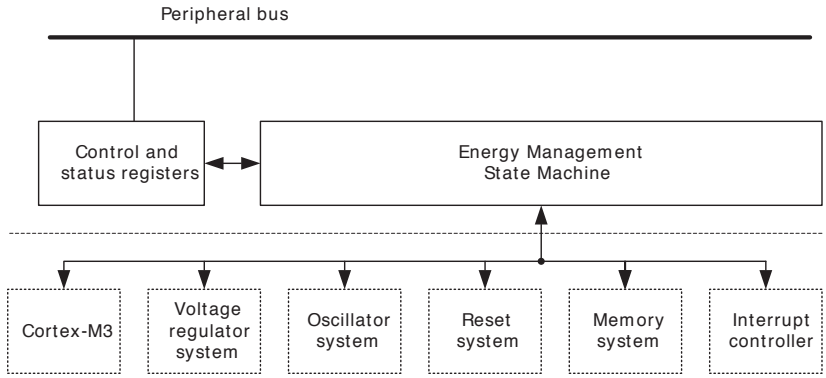


Figure 71: EMU Overview

The EMU is available as a peripheral on the peripheral bus. The energy management state machine is triggered from the Cortex-M3 and controls the internal voltage regulators, oscillators, memories and interrupt systems in the low energy modes. Events from the interrupt or reset systems can in turn cause the energy management state machine to return to its active state. This is further described in the following sections.

K.3.1 Energy Modes

There are five main energy modes available in the ARM core, called Energy Mode 0 (EM0) through Energy Mode 4 (EM4). EM0, also called the active mode, is the energy mode in which any peripheral function can be enabled and the Cortex-M3 core is executing instructions. EM1 through EM4, also called low energy modes, provide a selection of reduced peripheral functionality that also lead to reduced energy consumption, as described below.

Figure 72 shows the transitions between different energy modes. After reset the EMU will always start in EM0. A transition from EM0 to another energy mode is always initiated by software. EM0 is the highest activity mode, in which all functionality is available. EM0 is therefore also the mode with highest energy consumption.

The low energy modes EM1 through EM4 result in less functionality being available, and therefore also reduced energy consumption. The Cortex-M3 is not executing instructions in any low energy mode. Each low energy mode provides different energy consumptions associated with it, for example because a different set of peripherals are enabled or because these peripherals are configured differently.

A transition from EM0 to a low energy mode can only be triggered by software.

A transition from EM1 – EM3 to EM0 can be triggered by an enabled interrupt or event. In addition, a chip reset will return the device to EM0. A transition from EM4 can be triggered by a pin reset, power-on reset, EM4 GPIO wakeup, or Backup RTC interrupt.

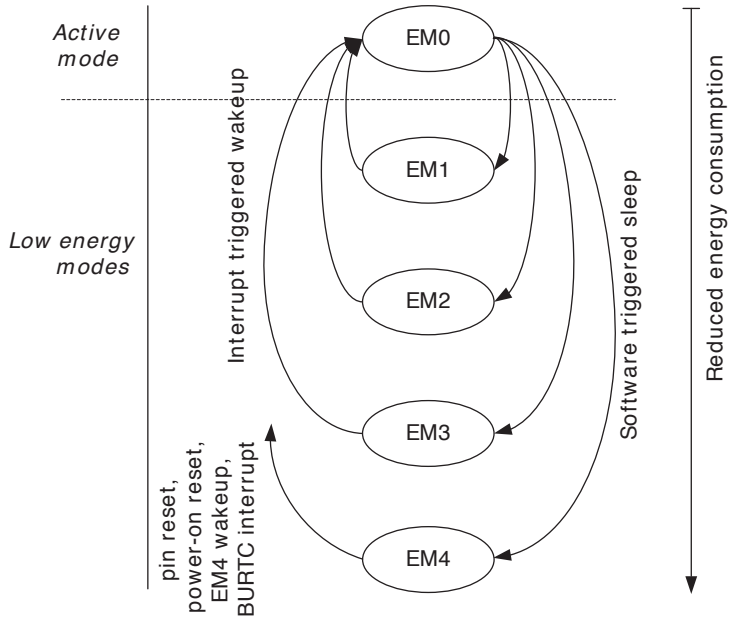


Figure 72: EMU Energy Mode Transitions

No direct transitions between EM1, EM2 or EM3 are available, as can also be seen from Figure 72. Instead, a wakeup will transition back to EM0, in which software can enter any other low energy mode. An overview of the supported energy modes and the functionality available in each mode is shown in Table 73. Most peripheral functionality indicated as "On" in a particular energy mode can also be turned off from software in order to save further energy.

The different Energy Modes are summarized in the following sections.

EM0

- The high frequency oscillator is active
- High frequency clock trees are active
- All peripheral functionality is available

EM1

- The high frequency oscillator is active
- MCU clock tree is inactive

[8] Energy Mode 0/Active Mode
[9] Energy Mode 1/2/3/4
[10] When the 1 kHz ULFRCO is selected
[11] Not available in Backup mode

	EM0[8]	EM1[9]	EM2[9]	EM3[9]	EM4[9]
Wakeup time to EM0	-	-	2 μs	2 μs	160 μs
MCU clock tree	On	-	-	-	-
High frequency peripheral clock trees	On	On	-	-	-
Core voltage regulator	On	On	-	-	-
High frequency oscillator	On	On	-	-	-
I^2C full functionality	On	On	-	-	-
Low frequency peripheral clock trees	On	On	On	-	-
Low frequency oscillator	On	On	On	-	-
Real Time Counter	On	On	On	On[10]	-
LEUART	On	On	On	-	-
LETIMER	On	On	On	On[10]	-
LESENSE	On	On	On	On[10]	-
PCNT	On	On	On	On	-
ACMP	On	On	On	On	-
I^2C receive address recognition	On	On	On	On	-
Watchdog	On	On	On	On[10]	-
Pin interrupts	On	On	On	On	-
RAM voltage regulator/RAM retention	On	On	On	On	-
Brown Out Reset	On	On	On	On	-
Power On Reset	On	On	On	On	On
Pin Reset	On	On	On	On	On
GPIO state retention	On	On	On	On	On[11]
EM4 Reset Wakeup Request	-	-	-	-	On[11]
Backup RTC	On	On	On	On	On
Backup retention registers	On	On	On	On	On

Figure 73: EMU Energy Mode Overview

- High frequency peripheral clock trees are active
- All peripheral functionality is available

EM2

- The high frequency oscillator is inactive
- The high frequency peripheral and MCU clock trees are inactive
- The low frequency oscillator and clock trees are active
- Low frequency peripheral functionality is available
- Wakeup through peripheral interrupt or asynchronous pin interrupt
- RAM and register values are preserved
- DAC and OPAMPs are available

EM3

- Both high and low frequency oscillators and clock trees are inactive
- Wakeup through asynchronous pin interrupts, I^2C address recognition or ACMP edge interrupt

- Watchdog and some low frequency peripherals available when ULFRCO (1 kHz clock) has been selected
- BURTC is available.
- All other peripheral functionality is disabled
- RAM and register values are preserved
- DAC and OPAMPs are available

EM4

- All oscillators and regulators are inactive, if Backup RTC is not enabled.
- RAM and register values are not preserved, except for the ones located in the Backup RTC.
- Optional GPIO state retention
- Wakeup from Backup RTC interrupt, external pin reset or pins that support EM4 wakeup

K.3.2 Entering a Low Energy Mode

A low energy mode is entered by first configuring the desired Energy Mode through the EMU_CTRL register and the SLEEPDEEP bit in the Cortex-M3 System Control Register, see Table 74. A Wait For Interrupt (WFI) or Wait For Event (WFE) instruction from the Cortex-M3 triggers the transition into a low energy mode.

The transition into a low energy mode can optionally be delayed until the lowest priority Interrupt Service Routine (ISR) is exited, if the SLEEPONEXIT bit in the Cortex-M3 System Control Register is set.

Entering the lowest energy mode, EM4, is done by writing a sequence to the EM4CTRL bitfield in the EMU_CTRL register. Writing a zero to the EM4CTRL bitfield will restart the power sequence. EM2BLOCK prevents the EMU to enter EM2 or lower, and it will instead enter EM1.

EM3 is equal to EM2, except that the LFACLK/LFBCLK are disabled in EM3. The LFACLK/LFBCLK must be disabled by the user before entering low energy mode.

The EMVREG bit in EMU_CTRL can be used to prevent the voltage regulator from being turned off in low energy modes. The device will then essentially stay in EM1 when entering a low energy mode.

Figure 74: EMU Entering a Low Energy Mode

Low Energy Mode	EM4CTRL	EMVREG	EM2BLOCK	SLEEPDEEP	Cortex-M3 Instruction
EM1	0	x	x	0	WFI or WFE
EM2	0	0	0	1	WFI or WFE
EM4	Write sequence: 2, 3, 2, 3, 2, 3, 2, 3, 2	x	x	x	x

('x' means don't care)

K.3.3 Leaving a Low Energy Mode

In each low energy mode a selection of peripheral units are available, and software can either enable or disable the functionality. Enabled interrupts that can cause wakeup from a low energy mode are shown in Table 75. The wakeup triggers always return the EFM32 to EM0. Additionally, any reset source will return to EM0.

Peripheral	Wakeup Trigger	EM0[12]	EM1[13]	EM2[13]	EM3[13]	EM4[13]
RTC	Any enabled interrupt	-	Yes	Yes	Yes[14]	-
USART	Receive / transmit	-	Yes	-	-	-
UART	Receive / transmit	-	Yes	-	-	-
LEUART	Receive / transmit	-	Yes	Yes	-	-
LESENSE	Any enabled interrupt	-	Yes	Yes	Yes[14]	-
I^2C	Any enabled interrupt	-	Yes	-	-	-
I^2C	Receive address recognition	-	Yes	Yes	Yes	-
TIMER	Any enabled interrupt	-	Yes	-	-	-
LETIMER	Any enabled interrupt	-	Yes	Yes	Yes[14]	-
CMU	Any enabled interrupt	-	Yes	-	-	-
DMA	Any enabled interrupt	-	Yes	-	-	-
MSC	Any enabled interrupt	-	Yes	-	-	-
DAC	Any enabled interrupt	-	Yes	-	-	-
ADC	Any enabled interrupt	-	Yes	-	-	-
AES	Any enabled interrupt	-	Yes	-	-	-
PCNT	Any enabled interrupt	-	Yes	Yes	Yes[15]	-
ACMP	Any enabled edge interrupt	-	Yes	Yes	Yes	-
VCMP	Any enabled edge interrupt	-	Yes	Yes	Yes	-
Pin interrupts	Asynchronous	-	Yes	Yes	Yes	-
Pin	Reset	-	Yes	Yes	Yes	Yes
EM4 wakeup on supported pins	Asynchronous	-	-	-	-	Yes
Backup RTC	Any enabled interrupt	Yes	Yes	Yes	Yes	Yes
Power	Cycle Off/On		Yes	Yes	Yes	Yes

Figure 75: EMU Wakeup Triggers from Low Energy Modes

K.3.4 Backup power domain

The ARM core can be partly powered by a backup battery. The backup power input, BU_VIN, is connected to a power domain in the ARM core containing the Backup RTC and 512 bytes of data retention, available in all energy modes. Figure 76 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

[12] Energy Mode 0/Active Mode
[13] Energy mode 1/2/3/4
[14] When the 1 kHz ULFRCO is selected
[15] When using an external clock

mode, equivalent to EM4, and automatically switches over to the backup power supply.

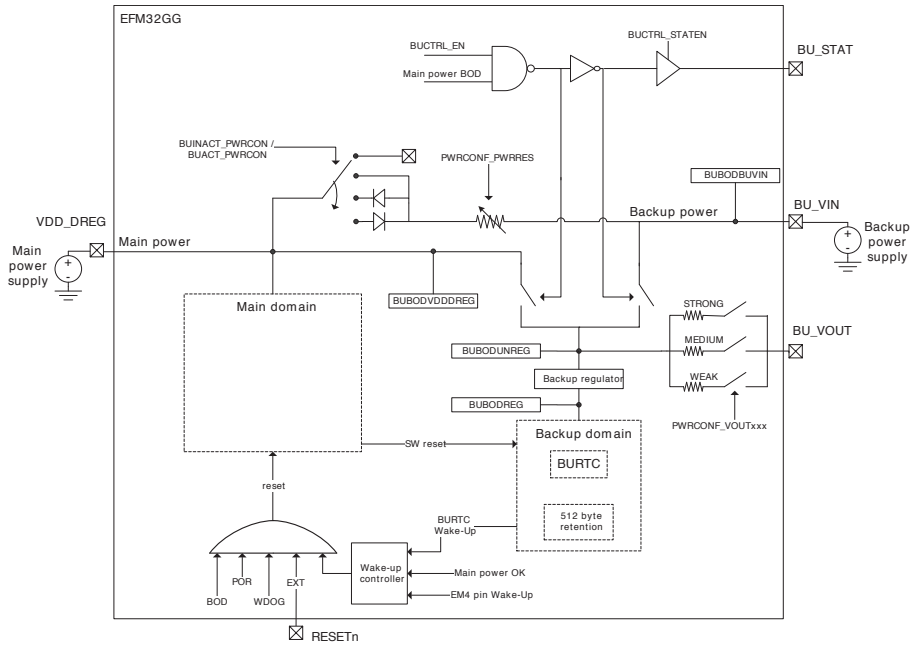


Figure 76: Backup power domain overview

When in backup mode, available functionality is the same as the functionality available in EM4. Refer to Section K.3.4 for further details.

Brown out detectors

The backup power domain functionality utilizes four brown-out detectors, BODs. One senses the main power supply, one senses the backup power supply, one senses the unregulated selected power supply (main or backup, depending on mode), and one BOD senses the regulated power supply. The bits BUBODVDDDREG ,BUBODBUVIN, BUBODUNREG, and BUBODREG in the RSTCAUSE register in the RMU are set when the associated BOD triggers. The locations of the Backup BODs are indicated in Figure 76. A brown out on the main power supply will trigger a switch to the backup power supply if the backup functionality is enabled and the BOD sensing on the backup power supply has not triggered. The two other BODs are used for error indication and will only set the bits in RMU_RSTCAUSE if they are triggered.

Entering backup mode

To be able to enter backup mode, the EN bit in EMU_BUCTRL has to be set. The BURDY interrupt flag will be set as soon as the backup sensing module is operational. Status of the backup functionality is also available in the BURDY flag in the EMU_STATUS register. The BU_VIN pin also needs to be enabled. This is

done by setting the BUVINPEN bit in EMU_ROUTE. To enter backup mode, the voltage on VDD_DREG has to drop below the programmable threshold of the BOD sensing on this power. This threshold is programmed using BUENRANGE and BUENTHRES in EMU_BUINACT. BUENRANGE decides the voltage range for the BOD, while BUENTHRES is used for tuning of the BOD threshold. Refer to Section K.3.4 for details regarding BOD calibration.

BUVINPEN in EMU_ROUTE is by default set. If Backup mode is not to be used, this bit should be cleared.

The voltage on BU_VIN has to be above the threshold for the BOD sensing on BU_VIN to enter backup mode.

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, set STATEN in EMU_BUCTRL. The BU_STAT pin is driven to BU_VIN when backup mode is active and to ground otherwise.

Leaving backup mode

To exit backup mode, the voltage on VDD_DREG has to be above the threshold programmed in EMU_BUACT. BUEXRANGE decides the voltage range for backup mode exit, while BUEXTHRES is used for tuning. When leaving backup mode, a system reset is triggered, resetting everything except the backup domain. When backup mode has been active, the BURST bit in RMU_RSTCAUSE is set.

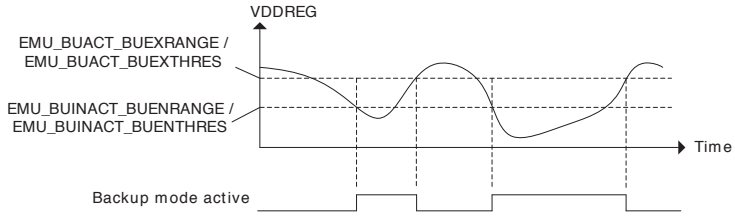


Figure 77: Entering and leaving backup mode

Figure 77 illustrates how the BOD sensing on VDD_DREG can be programmed to implement hysteresis on entering and exiting backup mode.

Threshold calibration

The thresholds for entering and exiting backup mode are configured in the EMU_BUINACT and EMU_BUACT registers, respectively. Calibration of these thresholds is performed during production test, but may also be performed using the DAC. The calibration values for the BODs sensing on unregulated power and BU_VIN, BUBODUNREG and BUBODBUVIN respectively, are available in EMU_BUBODVINCAL and EMU_BUBODUNREGCAL. These registers are written during production test and should not be modified except for calibrating the Backup BOD sensing on VDD_DREG, as described in the following section.

Setting BODCAL in EMU_BUCTRL will enable a mode where the BOD is sensing the DAC output, as depicted in Figure 78. For the BODCAL bit to take effect, the

backup power enable bit, EN in EMU_BUCTRL, has to be cleared. The procedure for BOD calibration is as follows:

- Clear EN and set BODCAL in EMU_BUCTRL.
- Store the values in EMU_BUBODVINCAL and EMU_BUBODUNREGCAL before clearing these registers.
- Configure the DAC to output to the maximum level and wait for 500 us before configuring the DAC output to the wanted BOD trigger voltage level.
- Step through the BOD calibration values (RANGE and THRES in EMU_BUINACT) with 500 us delay in between steps until the BUBODVDDDREG flag in RMU_RSTCAUSE is set. The RANGE and THRES values in EMU_BUINACT can now be written to EMU_BUINACT for configuration of threshold for entering backup mode, or EMU_BUACT for configuration of the threshold for leaving backup mode.
- Restore the values in EMU_BUBODVINCAL and EMU_BUBODUNREGCAL.

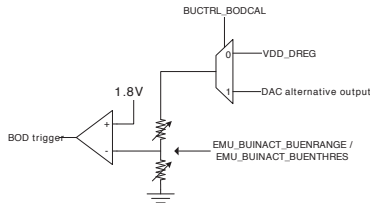


Figure 78: BOD calibration using DAC

Backup battery charging

The ARM core 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 PWRCON in EMU_BUINACT. PWRCON in EMU_BUACT configures the connection when in backup mode. The series resistance between the two power domains is configured in PWRRES in EMU_PWRCONF, this configuration applies both to backup mode and normal mode.

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. The switches are controlled using the VOUTWEAK, VOUTMED, and VOUTSTRONG bits in EMU_PWRCONF. For resistor values, refer to Device Datasheet Electrical Characteristics.

Voltage probing

It is possible to probe the voltage levels at VDD_DREG, BU_VIN, and BU_VOUT. This is done by configuring the ADC to measure a tristated channel, for instance a disabled DAC channel. The PROBE bitfield in EMU_BUCTRL configures which voltage

to be probed. The voltage measured by the ADC will be 1/8 of the actual probed voltage, meaning that the result needs to be multiplied by 8 for the correct result. Voltage probing does not work when BODCAL in the EMU_BUCTRL register is set.

Configuration lock

Configurations used in Backup mode and EM4, like BOD calibration, and Backup RTC settings need to be locked before entering EM4, this is done by setting the LOCKCONF bit in EMU_EM4CONF. This bit should also be set before a potential entry to backup mode. Setting this bit will lock following the configuration:

- LFXOMODE, LFXOBUFCUR, and LFXOBOOST in CMU_CTRL
- REDLFXOBOOST in EMU_AUXCTRL
- TUNING in CMU_LFRCOCTRL
- BURSTEN in RMU_CTRL
- BURTCWU and VREGEN in EMU_EM4CONF
- EMU_BUCTRL
- EMU_PWRCONF
- EMU_BUINACT
- EMU_BUACT
- EMU_ROUTE

For registers residing in the CMU and EMU_AUXCTRL, the reset value will be read after exit from EM4 or Backup mode, but if LOCKCONF in EMU_EM4CONF has been set, the locked configuration will be used until LOCKCONF is cleared. This also applies for the LOCKCONF bit itself.

EM4 with RTC and data retention

The backup power domain can also be powered by the main power. This provides possibility for Backup RTC operation and data retention in EM4. Available functionality in EM4 is configured in EMU_EM4CONF. Setting the VREGEN bit will keep the voltage regulator for the Backup domain enabled when in EM4. This allows the Backup RTC to keep running. To enable the Backup RTC to wake up the system from EM4, BURTCWU in EMU_EM4CONF needs to be set. When BURTCWU is set, any enabled Backup RTC interrupt will wake up the system. For further details regarding the Backup RTC and EM4 data retention, refer to [?].

The voltage regulator can also be used to power the Backup RTC during a watchdog reset from any energy mode. Set EMU_EM4CONF_VREGEN to enable the Backup RTC to be powered from the regulator, making sure it survives a watchdog reset.

Oscillators in EM4 When the system is in EM4 or backup mode with the voltage regulator enabled, the ULFRCO is by default enabled. If the LFXO or LFRCO is used

by the Backup RTC, the ULFRCO can be shut down to reduce power consumption. To do this, configure the OSC bitfield in EMU_EM4CONF.

If OSC in EMU_EM4CONF is not set to ULFRCO, PRESC and LPCOMP in BURTC_CTRL has to be configured in the following manner:

- 4 < (PRESC + LPCOMP) < 8, PRESC = 0,5,6,7

Refer to [?] for details on how to configure the Backup RTC.

Brown-out detector in EM4 To enable Brown-out detection in EM4, the Backup BODs have to be enabled, by setting EN in EMU_BUCTRL. When BURDY in EMU_STATUS is set, the Brown-out detectors are ready and able to issue a reset from EM4 if a Brown-out is detected on either regulated or unregulated power. The Backup BOD' ability to issue reset from EM4 can be disabled by setting BUBODRSTDIS in EMU_EM4CONF.

The Backup BODs can be enabled without allowing entrance to backup mode. This is done by setting EN in EMU_BUCTRL, and clearing BUVINPEN in EMU_ROUTE.

K.3.5 Powering off SRAM blocks

The SRAM blocks can be individually disabled using the POWERDOWN bitfield in the EMU_MEMCTRL register. To disable a block means that the power source is removed from the entire block, which will conserve energy. Once a block has been disabled it can only be enabled by reset.

All the blocks can be turned off except the first one.

K.4 Register Map

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

Offset	Name	Type	Description
0x000	EMU_CTRL	RW	Control Register
0x008	EMU_LOCK	RW	Configuration Lock Register
0x024	EMU_AUXCTRL	RW	Auxiliary Control Register
0x02C	EMU_EM4CONF	RW	Energy mode 4 configuration register
0x030	EMU_BUCTRL	RW	Backup Power configuration register
0x034	EMU_PWRCONF	RW	Power connection configuration register
0x038	EMU_BUINACT	RW	Backup mode inactive configuration register
0x03C	EMU_BUACT	RW	Backup mode active configuration register
0x040	EMU_STATUS	R	Status register
0x044	EMU_ROUTE	RW	I/O Routing Register
0x048	EMU_IF	R	Interrupt Flag Register
0x04C	EMU_IFS	W1	Interrupt Flag Set Register
0x050	EMU_IFC	W1	Interrupt Flag Clear Register
0x054	EMU_IEN	RW	Interrupt Enable Register
0x058	EMU_BUBODBUVINCAL	RW	BU_VIN Backup BOD calibration
0x05C	EMU_BUBODUNREGCAL	RW	Unregulated power Backup BOD calibration

K.5 Register Description

K.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		0	0
Access																													RW		RW	RW
Name																													EM4CTRL		EM2BLOCK	EMVREG

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3:2	EM4CTRL	0x0	RW	**Energy Mode 4 Control**
	This register is used to enter Energy Mode 4, in which the device only wakes up from an external pin reset, from a power cycle, Backup RTC interrupt, or EM4 wakeup reset request. Energy Mode 4 is entered when the EM4 sequence is written to this bitfield.			
1	EM2BLOCK	0	RW	**Energy Mode 2 Block**
	This bit is used to prevent the MCU to enter Energy Mode 2 or lower.			
0	EMVREG	0	RW	**Energy Mode Voltage Regulator Control**
	Control the voltage regulator in low energy modes 2 and 3.			

Value	Mode	Description
0	REDUCED	Reduced voltage regulator drive strength in EM2 and EM3.
1	FULL	Full voltage regulator drive strength in EM2 and EM3.

K.5.2 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																	RW															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	LOCKKEY	0x0000	RW	**Configuration Lock Key**
	Write any other value than the unlock code to lock all EMU registers, except the interrupt 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.

K.5.3 EMU_AUXCTRL - Auxiliary 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
Access																																RW
Name																																HRCCLR

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	HRCCLR	0	RW	**Hard Reset Cause Clear**
	Write to 1 and then 0 to clear the POR, BOD and WDOG reset cause register bits. See also the Reset Management Unit (RMU).			

K.5.4 EMU_EM4CONF - Energy mode 4 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												0	0x0		0	0
Access																RW												RW	RW		RW	RW
Name																LOCKCONF												BUBODRSTDIS	OSC		BURTCWU	VREGEN

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	LOCKCONF	0	RW	**EM4 configuration lock enable**
	Lock regulator, BOD and oscillator configuration. This is necessary before going to EM4 if the regulator is to be used in EM4, and must also be done before a potential entry to backup mode.			
15:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4	BUBODRSTDIS	0	RW	**Disable reset from Backup BOD in EM4**
	When set, no reset will be asserted due to Brownout when in EM4.			
3:2	OSC	0x0	RW	**Select EM4 duty oscillator**

Value	Mode	Description
0	ULFRCO	ULFRCO is available.
1	LFRCO	LFRCO is available. Can only be set if LFRCO is running before EM4/backup entry.
2	LFXO	LFXO is available. Can only be set if LFXO is available before EM4/backup entry.

1	BURTCWU	0	RW	**Backup RTC EM4 wakeup enable**
	Exit EM4 on Backup RTC interrupt.			
0	VREGEN	0	RW	**EM4 voltage regulator enable**
	When set, the voltage regulator is enabled in EM4, enabling operation of the Backup RTC and retention registers.			

K.5.5 EMU_BUCTRL - Backup Power configuration 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	0	0	0
Access																										RW			RW	RW	RW	RW
Name																										PROBE			BUMODEBODEN	BODCAL	STATEN	EN

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:5	PROBE	0x0	RW	**Voltage probe select**
	Configure which voltage to export to ADC.			

Bit	Name	Reset	Access	Description
	Value	**Mode**		**Description**
	0	DISABLE		Disable voltage probe.
	1	VDDDREG		Connect probe to VDD_DREG.
	2	BUIN		Connect probe to BU_IN.
	3	BUOUT		Connect probe to BU_OUT.
4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	BUMODEBODEN	0	RW	**Enable brown out detection on BU_VIN when in backup mode**
	When set, a reset (and switch back to main power) will be performed when in backup mode and the BUBODUNREG-bod senses a brown-out on BU_VIN.			
2	BODCAL	0	RW	**Enable BOD calibration mode**
	When set, the Backup BOD sensing on VDD_DREG will be sensing the DAC output.			
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.			

K.5.6 EMU_PWRCONF - Power connection configuration 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		0	0	0
Access																												RW		RW	RW	RW
Name																												PWRRES		VOUTSTRONG	VOUTMED	VOUTWEAK

Bit	Name	Reset	Access	Description
31:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4:3	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.
2	VOUTSTRONG	0	RW	**BU_VOUT strong enable**
	Enable strong switch between backup domain power supply and BU_VOUT.			
1	VOUTMED	0	RW	**BU_VOUT medium enable**
	Enable medium switch between backup domain power supply and BU_VOUT.			
0	VOUTWEAK	0	RW	**BU_VOUT weak enable**
	Enable weak switch between backup domain power supply and BU_VOUT.			

K.5.7 EMU_BUINACT - Backup mode inactive configuration 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		0x1		0x3		
Access																										RW		RW		RW		

Offset	Bit Position			
Name		PWRCON	BUENRANGE	BUENTHRES

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:5	PWRCON	0x0	RW	**Power connection configuration when not in Backup mode**

Value	Mode	Description
0	NONE	No connection.
1	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.
2	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.
3	NODIODE	Main power and backup power are connected without diode.

4:3	BUENRANGE	0x1	RW
	Threshold range for Backup BOD sensing on VDD_DREG when not in backup mode. This field is set to the threshold range calibrated during production, hence the reset value might differ from device to device.		
2:0	BUENTHRES	0x3	RW
	Threshold for Backup BOD sensing on VDD_DREG when not in backup mode. This field is set to the threshold value calibrated during production, hence the reset value might differ from device to device.		

K.5.8 EMU_BUACT - Backup mode active configuration register

Offset	Bit Position																														
0x03C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																										0x0		0x1		0x3		
Access																										RW		RW		RW		
Name																										PWRCON		BUEXRANGE		BUEXTHRES		

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:5	PWRCON	0x0	RW	**Power connection configuration when in Backup mode**

Value	Mode	Description
0	NONE	No connection.
1	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.
2	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.
3	NODIODE	Main power and backup power are connected without diode.

4:3	BUEXRANGE	0x1	RW
	Threshold range for Backup BOD sensing on VDD_DREG when in backup mode. This field is set to the threshold range calibrated during production, hence the reset value might differ from device to device.		
2:0	BUEXTHRES	0x3	RW
	Threshold for Backup BOD sensing on VDD_DREG when in backup mode. This field is set to the threshold value calibrated during production, hence the reset value might differ from device to device.		

K.5.9 EMU_STATUS - Status register

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

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	BURDY	0	R	**Backup mode ready**
	Set when the Backup power functionality is ready.			

K.5.10 EMU_ROUTE - I/O Routing 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																																1
Access																																RW
Name																																BUVINPEN

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	BUVINPEN	1	RW	**BU_VIN Pin Enable**
	When set, the BU_VIN pin is enabled.			

K.5.11 EMU_IF - Interrupt Flag Register

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

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	BURDY	0	R	**Backup functionality ready Interrupt Flag**
	Set when the Backup functionality is ready for use.			

K.5.12 EMU_IFS - Interrupt Flag Set 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
Access																																W1
Name																																BURDY

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	BURDY	0	W1	**Set Backup functionality ready Interrupt Flag**
	Write to 1 to set the BURDY interrupt flag.			

K.5.13 EMU_IFC - Interrupt Flag Clear 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
Access																																W1
Name																																BURDY

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	BURDY	0	W1	**Clear Backup functionality ready Interrupt Flag**
	Write to 1 to clear the BURDY interrupt flag.			

K.5.14 EMU_IEN - Interrupt Enable Register

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

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	BURDY	0	RW	**Backup functionality ready Interrupt Enable**
	Enable interrupt when Backup functionality is ready.			

K.5.15 EMU_BUBODBUVINCAL - BU_VIN Backup BOD calibration

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

Offset	Bit Position		
Reset		0x1	0x3
Access		RW	RW
Name		RANGE	THRES

Bit	Name	Reset	Access	Description
31:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4:3	RANGE	0x1	RW	
	Threshold range for Backup BOD sensing on BU_VIN. This field is set to the threshold range calibrated during production, hence the reset value might differ from device to device.			
2:0	THRES	0x3	RW	
	Threshold for Backup BOD sensing on BU_VIN. This field is set to the threshold value calibrated during production, hence the reset value might differ from device to device.			

K.5.16 EMU_BUBODUNREGCAL - Unregulated power Backup BOD calibration

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		0x3		
Access																												RW		RW		
Name																												RANGE		THRES		

Bit	Name	Reset	Access	Description
31:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4:3	RANGE	0x1	RW	
	Threshold range for Backup BOD sensing on unregulated power. This field is set to the threshold range calibrated during production, hence the reset value might differ from device to device.			
2:0	THRES	0x3	RW	
	Threshold for Backup BOD sensing on unregulated power. This field is set to the threshold value calibrated during production, hence the reset value might differ from device to device.			

L ARM Clock Management Unit

L.1 Introduction

The Clock Management Unit (CMU) is responsible for controlling the oscillators and clocks on-board the ARM core. 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 are inactive.

L.2 Features

- Multiple clock sources available:
  - 1-28 MHz High Frequency RC Oscillator (HFRCO)
  - 4-48 MHz High Frequency Crystal Oscillator (HFXO)
  - 32.768 Hz Low Frequency RC Oscillator (LFRCO)
  - 32.768 Hz Low Frequency Crystal Oscillator (LFXO)
  - 1 kHz Ultra Low Frequency RC Oscillator (ULFRCO)
- Low power oscillators
- Low start-up times
- Separate prescaler for High Frequency Core Clocks (HFCORECLK) and Peripheral Clocks (HFPERCLK)
- Individual clock prescaler selection for each Low Energy Peripheral
- Clock Gating on an individual basis to core modules and all peripherals
- Selectable clocks can be output on two pins for use externally.
- Auxiliary 1-28 MHz RC oscillator (AUXHFRCO) for flash programming, debug trace, and LESENSE timing.

L.3 Functional Description

An overview of the CMU is shown in Figure 79. The number of peripheral modules that are connected to the different clocks varies from device to device.

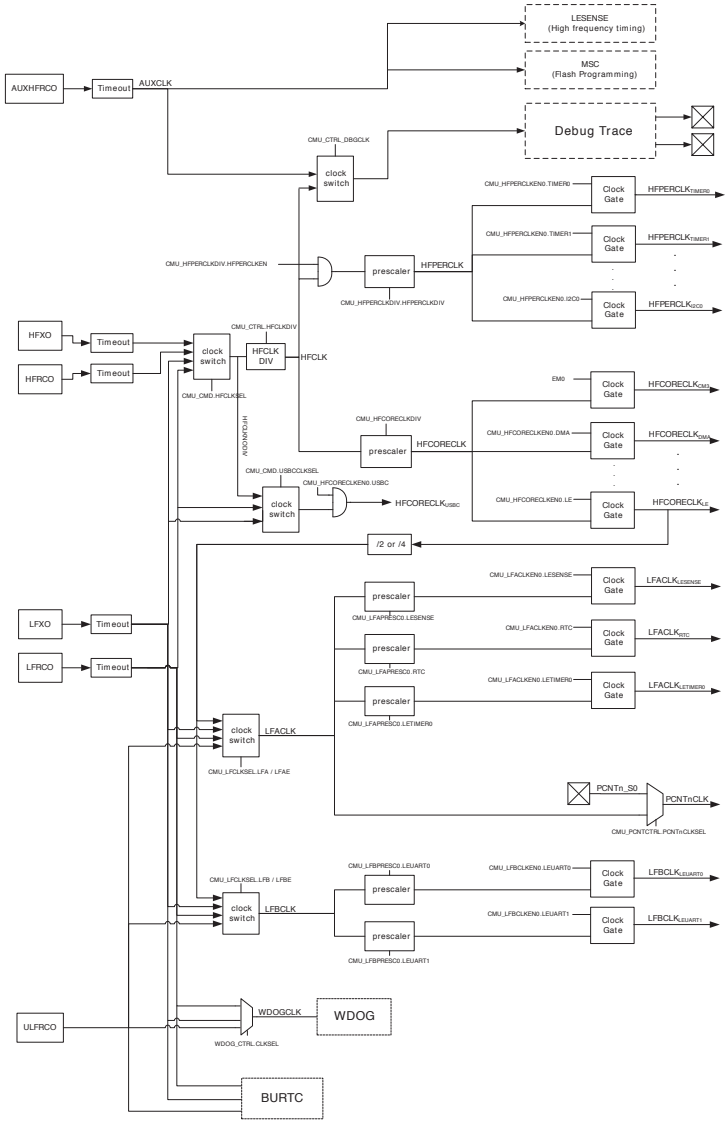


Figure 79: CMU Overview

L.3.1 System Clocks

HFCLK - High Frequency Clock

HFCLK is the selected High Frequency Clock. This clock is used by the CMU and drives the two prescalers that generate HFCORECLK and HFPERCLK. The HFCLK can be driven by a high-frequency oscillator (HFRCO or HFXO) or one of the low-

frequency oscillators (LFRCO or LFXO). By default the HFRCO is selected. In most applications, one of the high frequency oscillators will be the preferred choice. To change the selected HFCLK write to HFCLKSEL in CMU_CMD. The HFCLK is running in EM0 and EM1.

HFCLK can optionally be divided down by setting HFCLKDIV in CMU_CTRL to a nonzero value. This divides down HFCLK to all high frequency components except the USB Core and is typically used to save energy in USB applications where the system is not required to run at 48 MHz. Combined with the HFCORECLK and HFPERCLK prescalers the HFCLK divider also allows for more flexible clock division.

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. MSC, DMA etc. This also includes the interface to the Low Energy Peripherals. Some of the modules that are driven by this clock can be clock gated completely when not in use. This is done by clearing the clock enable bit for the specific module in CMU_HFCORECLKEN0. The frequency of HFCORECLK is set using the CMU_HFCORECLKDIV register. The setting can be changed dynamically and the new setting takes effect immediately.

The USB Core clock (USBC) is always undivided regardless of the HFCLKDIV setting. When the USB Core is active this clock must be switched to a 32 kHz clock (LFRCO or LFXO) when entering EM2. The USB Core uses this clock for monitoring the USB bus. The switch is done by writing USBCCLKSEL in CMU_CMD. The currently active clock can be checked by reading CMU_STATUS. The clock switch can take up to 1.5 32 kHz cycle (45 us). To avoid polling the clock selection status when switching switching from 32 kHz to HFCLK when coming up from EM2 the USBCHFCLKSEL interrupt can be used. EM3 is not supported when the USB is active.

If HFPERCLK runs faster than HFCORECLK, the number of clock cycles for each bus-access to peripheral modules will increase with the ratio between the clocks. Please refer to [?] for more details.

HFPERCLK - High Frequency Peripheral Clock

Like HFCORECLK, HFPERCLK can also be a prescaled version of HFCLK. This clock drives the High-Frequency Peripherals. All the peripherals that are driven by this clock can be clock gated completely when not in use. This is done by clearing the clock enable bit for the specific peripheral in CMU_HFPERCLKEN0. The frequency of HFPERCLK is set using the CMU_HFPERCLKDIV register. The setting can be changed dynamically and the new setting takes effect immediately.

If HFPERCLK 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 if HFPERCLK runs three times as fast as the HFCORECLK.

LFACLK - Low Frequency A Clock

LFACLK is the selected clock for the Low Energy A Peripherals. There are four selectable sources for LFACLK: LFRCO, LFXO, HFCORECLK/2 and ULFRCO. In addi-

tion, the LFACLK can be disabled. From reset, the LFACLK source is set to LFRCO. However, note that the LFRCO is disabled from reset. The selection is configured using the LFA field in CMU_LFCLKSEL. The HFCORECLK/2 setting allows the Low Energy A Peripherals to be used as high-frequency peripherals. If HFCORECLK/2 is selected as LFACLK, the clock will stop in EM2/3.

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.

LFBCLK - Low Frequency B Clock

LFBCLK is the selected clock for the Low Energy B Peripherals. There are four selectable sources for LFBCLK: LFRCO, LFXO, HFCORECLK/2 and ULFRCO. In addition, the LFBCLK can be disabled. From reset, the LFBCLK source is set to LFRCO. However, note that the LFRCO is disabled from reset. The selection is configured using the LFB field in CMU_LFCLKSEL. The HFCORECLK/2 setting allows the Low Energy B Peripherals to be used as high-frequency peripherals.

If HFCORECLK/2 is selected as LFBCLK, the clock will stop in EM2/3.

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.

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.

WDOGCLK - Watchdog Timer Clock

The Watchdog Timer (WDOG) can be configured to use one of three different clock sources: LFRCO, LFXO or ULFRCO. ULFRCO (Ultra Low Frequency RC Oscillator) is a separate 1 kHz RC oscillator that also runs in EM3.

AUXCLK - Auxiliary Clock

AUXCLK is a 1-28 MHz clock driven by a separate RC oscillator, AUXHFRCO. This clock is used for flash programming, and Serial Wire Output (SWO), and LESENSE operation. During flash programming, or if needed by LESENSE, this clock will be active. If the AUXHFRCO has not been enabled explicitly by software, the MSC or LESENSE module will automatically start and stop it. The AUXHFRCO is enabled by writing a 1 to AUXHFRCOEN in CMU_OSCENCMD. This explicit enabling is required when SWO is used.

L.3.2 Oscillator Selection

Start-up Time

The different oscillators have different start-up times. For the RC oscillators, the start-up time is fixed, but both the LFXO and the HFXO have configurable start-up time. At the end of the start-up time a ready flag is set to indicated that the start-up time has exceeded and that the clock is available. The low start-up time values can be used for an external clock source of already high quality, while the higher start-up times should be used when the clock signal is coming directly from a crystal. The startup time for HFXO and LFXO can be set by configuring the HFXOTIMEOUT and LFXOTIMEOUT bitfields, respectively. Both bitfields are located in CMU_CTRL. For HFXO it is also possible to enable a glitch detection filter by setting HFXOGLITCHDETEN in CMU_CTRL. The glitch detector will reset the start-up counter if a glitch is detected, making the start-up process start over again.

There are individual bits for each oscillator indicating the status of the oscillator:

- ENABLED - Indicates that the oscillator is enabled
- READY - Start-up time is exceeded
- SELECTED - Start-up time is exceeded and oscillator is chosen as clock source

These status bits are located in the CMU_STATUS register.

Switching Clock Source

The HFRCO oscillator is a low energy oscillator with extremely short wake-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 and EM3). After reset, the HFRCO frequency is 14 MHz.

Software can switch between the different clock sources at run-time. E.g., when the HFRCO is the clock source, software can switch to HFXO by writing the field HFCLKSEL in the CMU_CMD command register. See Figure 80 for a description of the sequence of events for this specific operation.

It is important first to enable the HFXO since switching to a disabled oscillator will effectively stop HFCLK and only a reset can recover the system.

During the start-up period HFCLK will stop 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 HFXO and then wait for the oscillator to become ready before switching the clock source. This way, the system continues to run on the HFRCO until the HFXO has timed out and provides a reliable clock. This sequence of events is shown in Figure 81.

A separate flag is set when the oscillator is ready. This flag can also be configured to generate an interrupt.

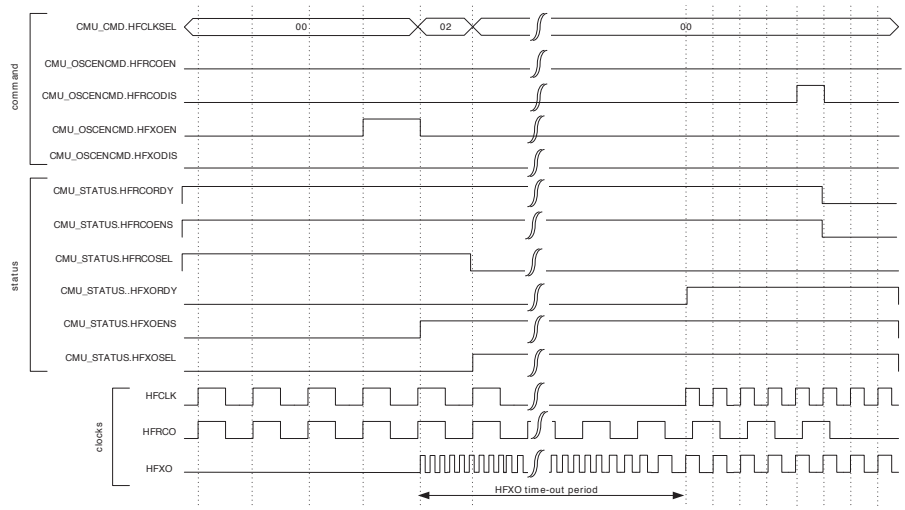


Figure 80: CMU Switching from HFRCO to HFXO before HFXO is ready

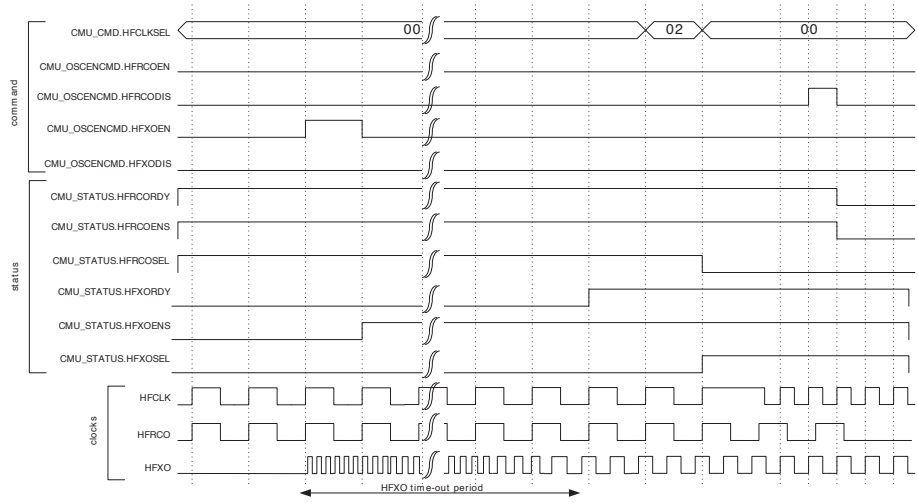


Figure 81: CMU Switching from HFRCO to HFXO after HFXO is ready

Switching clock source for LFACLK and LFBCLK is done by setting the LFA and LFB fields in CMU_LFCLKSEL. To ensure no stalls in the Low Energy Peripherals, the clock source should be ready before switching to it.

To save energy, remember to turn off all oscillators not in use.

L.3.3 Oscillator Configuration

HFXO and LFXO

The crystal oscillators are by default configured to ensure safe startup and operation of the most common crystals. In order to optimize startup margin, startup time and power consumption for a given crystal, it is possible to adjust the gain in the oscillator. HFXO gain can be increased by setting HFXOBOOST field in CMU_CTRL, LFXO gain can be increased by setting LFXOBOOST field in CMU_CTRL or reduced by setting REDLFXOBOOST field in EMU_AUXCTRL. It is important that the boost settings, along with the crystal load capacitors are matched to the crystals in use. Correct values for these parameters can be found using the energyAware Designer.

The HFXO crystal is connected to the HFXTAL_N/HFXTAL_P pins as shown in Figure 82

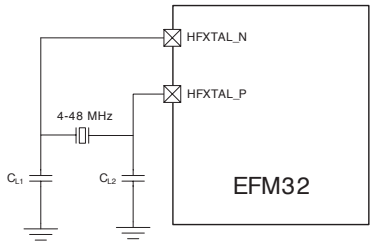


Figure 82: HFXO Pin Connection

Similarly, the LFXO crystal is connected to the LFXTAL_N/LFXTAL_P pins as shown in Figure 83

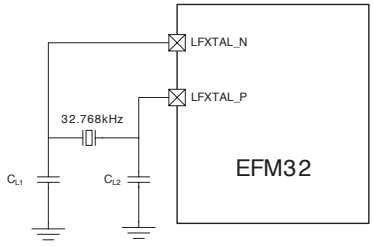


Figure 83: LFXO Pin Connection

It is possible to connect an external clock source to HFXTAL_N/LFXTAL_N pin of the HFXO or LFXO oscillator. By configuring the HFXOMODE/LFXOMODE fields in CMU_CTRL, the HFXO/LFXO can be bypassed.

HFRCO, LFRCO and AUXHFRCO

It is possible to calibrate the HFRCO, LFRCO and AUXHFRCO to achieve higher accuracy (see the device datasheets for details on accuracy). The frequency is adjusted by changing the TUNING fields in CMU_HFRCOCTRL/CMU_LFRCOCTRL/CMU_AUXHFRCOCTRL. Changing to a higher value will result in a higher frequency. Please refer to the datasheet for stepsize details.

The HFRCO and AUXHFRCO can be set to one of several different frequency bands from 1 MHz to 28 MHz by setting the BAND field in CMU_HFRCOCTRL and CMU_AUXHFRCOCTRL.The HFRCO and AUXHFRCO frequency bands are cali-

brated during production test, and the production tested calibration values can be read from the Device Information (DI) page. The DI page contains a separate tuning value for each frequency band. During reset, HFRCO and AUXHFRCO tuning values are set to the production calibrated values for the 14 MHz band, which is the default frequency band. When changing to a different HFRCO or AUXHFRCO band, make sure to also update the tuning value.

The LFRCO and is also calibrated in production and its TUNING value is set to the correct value during reset.

The CMU has built-in HW support to efficiently calibrate the RC oscillators at run-time, see Figure 84 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 smallest value that can be written to the CMU_CALCNT is 1. 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 at this point. 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. Then it is easy to find the ratio between the reference and the oscillator subject to the calibration. Overflows of the up-counter will not occur. If the up-counter reaches its top value before the down counter reaches 0, the top 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.

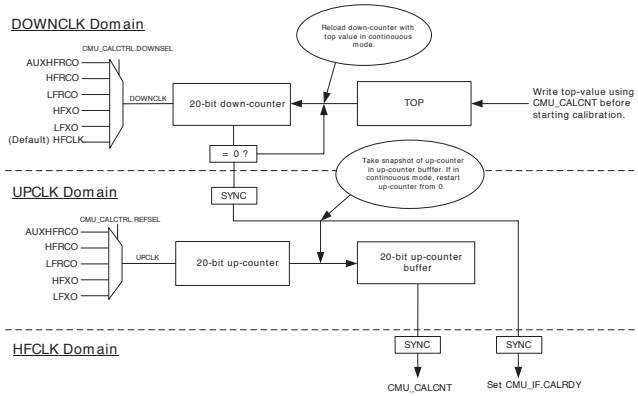


Figure 84: HW-support for RC Oscillator Calibration

The counter operation for single and continuous mode are shown in Figure 85 and Figure 86 respectively.

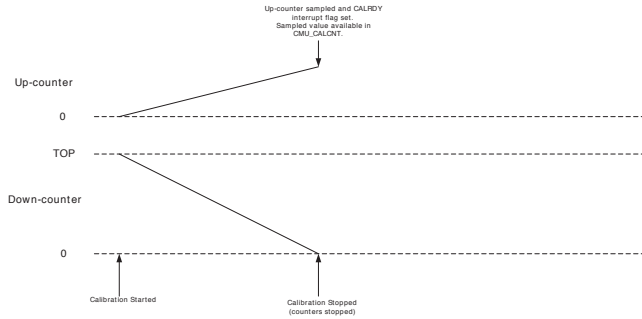


Figure 85: Single Calibration (CONT=0)

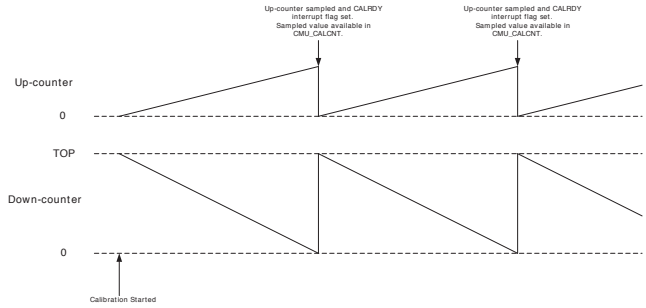


Figure 86: Continuous Calibration (CONT=1)

L.3.4 Configuration For Operating Frequencies

The HFXO is capable of driving crystals up to 48 MHz, which allows the EFM32 to run at up to this frequency. Different frequencies have different requirements as shown in Table 87. 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.

Maximum Frequency	MODE in MSC_READCTRL	HFLE in CMU_CTRL	HFXOBUFCUR in CMU_CTRL
16 MHz	WS0 / WS0SCBTP / WS1 / WS1SCBTP / WS2 / WS2SCBTP	-	BOOSTUPTO32MHZ (default value)
32 MHz	WS1 / WS1SCBTP / WS2 / WS2SCBTP	-	BOOSTUPTO32MHZ (default value)
48 MHz	WS2 / WS2SCBTP	1	BOOSTABOVE32MHZ

Figure 87: Configuration For Operating Frequencies

MODE in MSC_READCTRL makes sure the flash is able to operate at the given frequencies by inserting waitstates for flash accesses. HFXOBUFCUR in CMU_CTRL should be set to BOOSTABOVE32MHZ when operating above 32 MHz. When operating at 32 MHz or below, the default value (BOOSTUPTO32MHZ) should be used. HFLE in CMU_CTRL is only required for frequencies above 32 MHz, and ensures correct operation of LE peripherals. The CMU_CTRL_HFLE is or'ed with

HFCORECLKLEDIV in CMU_HFCORECLKDIV, so setting either of this bits will reduce the the frequency of CMU_HFCORECLKLEDIV2.

L.3.5 Output Clock on a Pin

It is possible to configure the CMU to output clocks on two pins. This clock selection is done using CLKOUTSEL0 and CLKOUTSEL1 fields in CMU_CTRL. The output pins must be configured in the CMU_ROUTE register.

- LFRCO, LFXO, HFCLK or the qualified clock from any of the oscillators can be output on one pin (CMU_OUT1). A qualified clock will not have any glitches or skewed duty-cycle during startup. For LFXO and HFXO you need to configure LFXOTIMEOUT and HFXOTIMEOUT in CMU_CTRL correctly to guarantee a qualified clock.
- HFRCO, HFXO, HFCLK/2, HFCLK/4, HFCLK/8, HFCLK/16, ULFRCO or AUXHFRCO can be output on another pin (CMU_OUT0)

Note that HFXO and HFRCO clock outputs to pin can be unstable after startup and should not be output on a pin before HFXORDY/HFRCORDY is set high in CMU_STATUS.

L.3.6 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.

L.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
0x004	CMU_HFCORECLKDIV	RW	High Frequency Core Clock Division Register
0x008	CMU_HFPERCLKDIV	RW	High Frequency Peripheral Clock Division Register
0x00C	CMU_HFRCOCTRL	RW	HFRCO Control Register
0x010	CMU_LFRCOCTRL	RW	LFRCO Control Register
0x014	CMU_AUXHFRCOCTRL	RW	AUXHFRCO Control Register
0x018	CMU_CALCTRL	RW	Calibration Control Register
0x01C	CMU_CALCNT	RWH	Calibration Counter Register
0x020	CMU_OSCENCMD	W1	Oscillator Enable/Disable Command Register
0x024	CMU_CMD	W1	Command Register
0x028	CMU_LFCLKSEL	RW	Low Frequency Clock Select Register
0x02C	CMU_STATUS	R	Status Register
0x030	CMU_IF	R	Interrupt Flag Register
0x034	CMU_IFS	W1	Interrupt Flag Set Register
0x038	CMU_IFC	W1	Interrupt Flag Clear Register
0x03C	CMU_IEN	RW	Interrupt Enable Register
0x040	CMU_HFCORECLKEN0	RW	High Frequency Core Clock Enable Register 0
0x044	CMU_HFPERCLKEN0	RW	High Frequency Peripheral Clock Enable Register 0
0x050	CMU_SYNCBUSY	R	Synchronization Busy Register
0x054	CMU_FREEZE	RW	Freeze Register
0x058	CMU_LFACLKEN0	RW	Low Frequency A Clock Enable Register 0 (Async Reg)
0x060	CMU_LFBCLKEN0	RW	Low Frequency B Clock Enable Register 0 (Async Reg)
0x068	CMU_LFAPRESC0	RW	Low Frequency A Prescaler Register 0 (Async Reg)
0x070	CMU_LFBPRESC0	RW	Low Frequency B Prescaler Register 0 (Async Reg)

Offset	Name	Type	Description
0x078	CMU_PCNTCTRL	RW	PCNT Control Register
0x080	CMU_ROUTE	RW	I/O Routing Register
0x084	CMU_LOCK	RW	Configuration Lock Register

L.5 Register Description

L.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		0		0			0x0			0x0			0x3		0	0x0			1	0x0		0x3			0	0x1			0x3		0x0	
Access		RW		RW			RW			RW			RW		RW	RW			RW	RW		RW			RW	RW			RW		RW	
Name		HFLE		DBGCLK			CLKOUTSEL1			CLKOUTSEL0			LFXOTIMEOUT		LFXOBUFCUR	HFCLKDIV			LFXOBOOST	LFXOMODE		HFXOTIMEOUT			HFXOGLITCHDETEN	HFXOBUFCUR			HFXOBOOST		HFXOMODE	

Bit	Name	Reset	Access	Description
31	Reserved	To ensure compatibility with future devices, always write bits to 0.		
30	HFLE	0	RW	**High-Frequency LE Interface**
	Set to allow access to LE peripherals when running at frequencies higher than 24 MHz. Or'ed with CMU_HFCORECLKDIV_HFCORECLKLEDIV to reduce the frequency of CMU_HFCORECLKLEDIV2.			
29	Reserved	To ensure compatibility with future devices, always write bits to 0.		
28	DBGCLK	0	RW	**Debug Clock**
	Select clock used for the debug system.			

Value	Mode	Description
0	AUXHFRCO	AUXHFRCO is the debug clock.
1	HFCLK	The system clock is the debug clock.

Bit	Name	Reset	Access	Description
27:26	Reserved	To ensure compatibility with future devices, always write bits to 0.		
25:23	CLKOUTSEL1	0x0	RW	**Clock Output Select 1**
	Controls the clock output multiplexer. To actually output on the pin, set CLKOUT1PEN in CMU_ROUTE.			

Value	Mode	Description
0	LFRCO	LFRCO (directly from oscillator).
1	LFXO	LFXO (directly from oscillator).
2	HFCLK	HFCLK.
3	LFXOQ	LFXO (qualified).
4	HFXOQ	HFXO (qualified).
5	LFRCOQ	LFRCO (qualified).
6	HFRCOQ	HFRCO (qualified).
7	AUXHFRCOQ	AUXHFRCO (qualified).

Bit	Name	Reset	Access	Description
22:20	CLKOUTSEL0	0x0	RW	**Clock Output Select 0**
	Controls the clock output multiplexer. To actually output on the pin, set CLKOUT0PEN in CMU_ROUTE.			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	HFRCO	HFRCO (directly from oscillator).
1	HFXO	HFXO (directly from oscillator).
2	HFCLK2	HFCLK/2.
3	HFCLK4	HFCLK/4.
4	HFCLK8	HFCLK/8.
5	HFCLK16	HFCLK/16.
6	ULFRCO	ULFRCO (directly from oscillator).
7	AUXHFRCO	AUXHFRCO (directly from oscillator).

19:18 LFXOTIMEOUT 0x3 RW **LFXO Timeout**

Configures the start-up delay for LFXO.

Value	Mode	Description
0	8CYCLES	Timeout period of 8 cycles.
1	1KCYCLES	Timeout period of 1024 cycles.
2	16KCYCLES	Timeout period of 16384 cycles.
3	32KCYCLES	Timeout period of 32768 cycles.

17 LFXOBUFCUR 0 RW **LFXO Boost Buffer Current**

This value has been updated to the correct level during calibration and should not be changed.

16:14 HFCLKDIV 0x0 RW **HFCLK Division**

Use to divide HFCLK frequency by (HFCLKDIV + 1).

13 LFXOBOOST 1 RW **LFXO Start-up Boost Current**

Adjusts start-up boost current for LFXO.

Value	Mode	Description
0	70PCENT	70 %.
1	100PCENT	100 %.

12:11 LFXOMODE 0x0 RW **LFXO Mode**

Set this to configure the external source for the LFXO. 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	32.768 kHz crystal oscillator.
1	BUFEXTCLK	An AC coupled buffer is coupled in series with LFXTAL_N pin, suitable for external sinus wave (32.768 kHz).
2	DIGEXTCLK	Digital external clock on LFXTAL_N pin. Oscillator is effectively bypassed.

10:9 HFXOTIMEOUT 0x3 RW **HFXO Timeout**

Configures the start-up delay for HFXO.

Value	Mode	Description
0	8CYCLES	Timeout period of 8 cycles.
1	256CYCLES	Timeout period of 256 cycles.
2	1KCYCLES	Timeout period of 1024 cycles.
3	16KCYCLES	Timeout period of 16384 cycles.

8 Reserved To ensure compatibility with future devices, always write bits to 0.

7 HFXOGLITCHDETEN 0 RW **HFXO Glitch Detector Enable**

This bit enables the glitch detector which is active as long as the start-up ripple-counter is counting. A detected glitch will reset the ripple-counter effectively increasing the start-up time. Once the ripple-counter has timed-out, glitches will not be detected.

6:5 HFXOBUFCUR 0x1 RW **HFXO Boost Buffer Current**

The current level in the HFXO buffer should be set to default value when operating on 32 MHz or below. When operating on frequencies above 32 MHz, the buffer current level should be set to 3.

Value	Mode	Description
1	BOOSTUPTO32MHZ	Boost Buffer Current level when HFXO is below or equal to 32 MHz.
3	BOOSTABOVE32MHZ	Boost Buffer Current Level when HFXO is above 32 MHz.

4 Reserved To ensure compatibility with future devices, always write bits to 0.

3:2 HFXOBOOST 0x3 RW **HFXO Start-up Boost Current**

Used to adjust start-up boost current for HFXO.

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	50PCENT	50 %.
1	70PCENT	70 %.
2	80PCENT	80 %.
3	100PCENT	100 % (default).

1:0 HFXOMODE 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. The oscillator setting is reset to default when 1 is written to HFXODIS in CMU_OSCENCMD.

Value	Mode	Description
0	XTAL	4-48 MHz crystal oscillator.
1	BUFEXTCLK	An AC coupled buffer is coupled in series with HFXTAL_N, suitable for external sine wave (4-48 MHz). The sine wave should have a minimum of 200 mV peak to peak.
2	DIGEXTCLK	Digital external clock on HFXTAL_N pin. Oscillator is effectively bypassed.

L.5.2 CMU_HFCORECLKDIV - High Frequency Core Clock Division 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					0x0			
Access																								RW					RW			
Name																								HFCORECLKLEDIV					HFCORECLKDIV			

Bit	Name	Reset	Access	Description
31:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8	HFCORECLKLEDIV	0	RW	**Additional Division Factor For HFCORECLKLE**

Additional division factor for HFCORECLKLE. When running at frequencies higher than 24 MHz, this must be set to DIV4.

Value	Mode	Description
0	DIV2	Valid for frequencies 24 MHz and lower.
1	DIV4	Must be used when HFCORECLK may go above 24 MHz.

7:4 Reserved To ensure compatibility with future devices, always write bits to 0.

3:0 HFCORECLKDIV 0x0 RW **HFCORECLK Divider**

Specifies the clock divider for HFCORECLK.

Value	Mode	Description
0	HFCLK	HFCORECLK = HFCLK.
1	HFCLK2	HFCORECLK = HFCLK/2.
2	HFCLK4	HFCORECLK = HFCLK/4.
3	HFCLK8	HFCORECLK = HFCLK/8.
4	HFCLK16	HFCORECLK = HFCLK/16.
5	HFCLK32	HFCORECLK = HFCLK/32.
6	HFCLK64	HFCORECLK = HFCLK/64.
7	HFCLK128	HFCORECLK = HFCLK/128.
8	HFCLK256	HFCORECLK = HFCLK/256.
9	HFCLK512	HFCORECLK = HFCLK/512.

L.5.3 CMU_HFPERCLKDIV - High Frequency Peripheral Clock Division 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					0x0			
Access																								RW					RW			
Name																								HFPERCLKEN					HFPERCLKDIV			

Bit	Name	Reset	Access	Description
31:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8	HFPERCLKEN	1	RW	**HFPERCLK Enable**
	Set to enable the HFPERCLK.			
7:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3:0	HFPERCLKDIV	0x0	RW	**HFPERCLK Divider**

Specifies the clock divider for the HFPERCLK.

Value	Mode	Description
0	HFCLK	HFPERCLK = HFCLK.
1	HFCLK2	HFPERCLK = HFCLK/2.
2	HFCLK4	HFPERCLK = HFCLK/4.
3	HFCLK8	HFPERCLK = HFCLK/8.
4	HFCLK16	HFPERCLK = HFCLK/16.
5	HFCLK32	HFPERCLK = HFCLK/32.
6	HFCLK64	HFPERCLK = HFCLK/64.
7	HFCLK128	HFPERCLK = HFCLK/128.
8	HFCLK256	HFPERCLK = HFCLK/256.
9	HFCLK512	HFPERCLK = HFCLK/512.

L.5.4 CMU_HFRCOCTRL - HFRCO 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																0x00						0x3			0x80							
Access																RW						RW			RW							
Name																SUDELAY						BAND			TUNING							

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16:12	SUDELAY	0x00	RW	**HFRCO Start-up Delay**
	Always write this field to 0.			
11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	BAND	0x3	RW	**HFRCO Band Select**

Write this field to set the frequency band in which the HFRCO is to operate. 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. To ensure an accurate frequency, the HFTUNING value should also be writtenwhen changing the frequency band. The calibrated tuning value for the different bands can be read from the Device Information page.

Value	Mode	Description
0	1MHZ	1 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
1	7MHZ	7 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
2	11MHZ	11 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
3	14MHZ	14 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
4	21MHZ	21 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
5	28MHZ	28 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.

Bit	Name	Reset	Access	Description
7:0	TUNING	0x80	RW	**HFRCO Tuning Value**
	Writing this field adjusts the HFRCO frequency (the higher value, the higher frequency). This field is updated with the production calibrated value for the 14 MHz band during reset, and the reset value might therefore vary between devices.			

L.5.5 CMU_LFRCOCTRL - LFRCO 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																										0x40						
Access																										RW						
Name																										TUNING						

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:0	TUNING	0x40	RW	**LFRCO Tuning Value**
	Writing this field adjusts the LFRCO frequency (the higher value, the higher frequency). This field is updated with the production calibrated value during reset, and the reset value might therefore vary between devices.			

L.5.6 CMU_AUXHFRCOCTRL - AUXHFRCO 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																						0x0			0x80							
Access																						RW			RW							
Name																						BAND			TUNING							

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	BAND	0x0	RW	**AUXHFRCO Band Select**
	Write this field to set the frequency band in which the AUXHFRCO is to operate. When changing this setting there will be no glitches on the AUXHFRCO output,hence it is safe to change this setting even while the system is using the AUXHFRCO. To ensure an accurate frequency, the AUXTUNING value should also be writtenwhen changing the frequency band. The calibrated tuning value for the different bands can be read from the Device Information page. Flash erase and write use this clock. If it is changed to another value than the default, MSC_TIMEBASE must also be configured to ensure correct flash erase and write operation.			

Value	Mode	Description
0	14MHZ	14 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
1	11MHZ	11 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
2	7MHZ	7 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
3	1MHZ	1 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
6	28MHZ	28 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.
7	21MHZ	21 MHz band. NOTE: Also set the TUNING value (bits 7:0) when changing band.

7:0	TUNING	0x80	RW	**AUXHFRCO Tuning Value**
	Writing this field adjusts the AUXHFRCO frequency (the higher value, the higher frequency).This field is updated with the production calibrated value during reset, and the reset value might therefore vary between devices.			

L.5.7 CMU_CALCTRL - Calibration 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	0x0			0x0		
Access																										RW	RW			RW		
Name																										CONT	DOWNSEL			UPSEL		

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6	CONT	0	RW	**Continuous Calibration**
	Set this bit to enable continuous calibration.			
5:3	DOWNSEL	0x0	RW	**Calibration Down-counter Select**
	Selects clock source for the calibration down-counter.			

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.

Bit	Name	Reset	Access	Description
2:0	UPSEL	0x0	RW	**Calibration Up-counter Select**
	Selects clock source for the calibration up-counter.			

Value	Mode	Description
0	HFXO	Select HFXO as up-counter.
1	LFXO	Select LFXO as up-counter.
2	HFRCO	Select HFRCO as up-counter.
3	LFRCO	Select LFRCO as up-counter.
4	AUXHFRCO	Select AUXHFRCO as up-counter.

L.5.8 CMU_CALCNT - Calibration Counter 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													0x00000																			
Access													RWH																			
Name													CALCNT																			

Bit	Name	Reset	Access	Description
31:20	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

L.5.9 CMU_OSCENCMD - Oscillator Enable/Disable Command 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																							W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1

Offset	Bit Position									
Name	LFXODIS	LFXOEN	LFRCODIS	LFRCOEN	AUXHFRCODIS	AUXHFRCOEN	HFXODIS	HFXOEN	HFRCODIS	HFRCOEN

Bit	Name	Reset	Access	Description
31:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9	LFXODIS	0	W1	**LFXO Disable**
	Disables the LFXO. LFXOEN has higher priority if written simultaneously.			
8	LFXOEN	0	W1	**LFXO Enable**
	Enables the LFXO.			
7	LFRCODIS	0	W1	**LFRCO Disable**
	Disables the LFRCO. LFRCOEN has higher priority if written simultaneously.			
6	LFRCOEN	0	W1	**LFRCO Enable**
	Enables the LFRCO.			
5	AUXHFRCODIS	0	W1	**AUXHFRCO Disable**
	Disables the AUXHFRCO. AUXHFRCOEN has higher priority if written simultaneously. WARNING: Do not disable this clock during a flash erase/write operation.			
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 HFRXO if this oscillator is selected as the source for HFCLK.			
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.			

L.5.10 CMU_CMD - Command 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	0	0x0		
Access																										W1		W1	W1	W1		
Name																										USBCCLKSEL		CALSTOP	CALSTART	HFCLKSEL		

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:5	USBCCLKSEL	0x0	W1	**USB Core Clock Select**
	Selects the clock for $HFCORECLK_{USBC}$. The status register is updated when the clock switch has taken effect.			

Value	Mode	Description
1	HFCLKNODIV	Select HFCLKNODIV as $HFCORECLK_{USBC}$.
2	LFXO	Select LFXO as $HFCORECLK_{USBC}$.
3	LFRCO	Select LFRCO as $HFCORECLK_{USBC}$.

4	CALSTOP	0	W1	**Calibration Stop**

Bit	Name	Reset	Access	Description
	Stops the calibration counters.			
3	CALSTART	0	W1	**Calibration Start**
	Starts the calibration, effectively loading the CMU_CALCNT into the down-counter and start decrementing.			
2:0	HFCLKSEL	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.

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.

L.5.11 CMU_LFCLKSEL - Low Frequency Clock Select 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													0x1		0x1	
Access												RW				RW													RW		RW	
Name												LFBE				LFAE													LFB		LFA	

Bit	Name	Reset	Access	Description
31:21	Reserved	To ensure compatibility with future devices, always write bits to 0.		
20	LFBE	0	RW	**Clock Select for LFB Extended**

This bit redefines the meaning of the LFB field.

Value	Mode	Description
0	DISABLED	LFBCLK is disabled (when LFB = DISABLED).
1	ULFRCO	ULFRCO selected as LFBCLK (when LFB = DISABLED).

19:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	LFAE	0	RW	**Clock Select for LFA Extended**

This bit redefines the meaning of the LFA field.

Value	Mode	Description
0	DISABLED	LFACLK is disabled (when LFA = DISABLED).
1	ULFRCO	ULFRCO selected as LFACLK (when LFA = DISABLED).

15:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3:2	LFB	0x1	RW	**Clock Select for LFB**

Selects the clock source for LFBCLK.

LFB	LFBE	Mode	Description
0	0	Disabled	LFBCLK is disabled
1	0	LFRCO	LFRCO selected as LFBCLK
2	0	LFXO	LFXO selected as LFBCLK
3	0	HFCORECLKLEDIV2	$HFCORECLK_{LE}$ divided by two is selected as LFBCLK
0	1	ULFRCO	ULFRCO selected as LFBCLK

1:0	LFA	0x1	RW	**Clock Select for LFA**

Selects the clock source for LFACLK.

LFA	LFAE	Mode	Description
0	0	Disabled	LFACLK is disabled
1	0	LFRCO	LFRCO selected as LFACLK
2	0	LFXO	LFXO selected as LFACLK
3	0	HFCORECLKLEDIV2	$HFCORECLK_{LE}$ divided by two is selected as LFACLK
0	1	ULFRCO	ULFRCO selected as LFACLK

L.5.12 CMU_STATUS - Status Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset															0	0	0	0	0	0	0	1	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
Name															USBCLFRCOSEL	USBCLFXOSEL	USBCHFCLKSEL	CALBSY	LFXOSEL	LFRCOSEL	HFXOSEL	HFRCOSEL	LFXORDY	LFXOENS	LFRCORDY	LFRCOENS	AUXHFRCORDY	AUXHFRCOENS	HFXORDY	HFXOENS	HFRCORDY	HFRCOENS

Bit	Name	Reset	Access	Description
31:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17	USBCLFRCOSEL	0	R	**USBC LFRCO Selected**
	LFRCO is selected (and active) as $HFCORECLK_{USBC}$.			
16	USBCLFXOSEL	0	R	**USBC LFXO Selected**
	LFXO is selected (and active) as $HFCORECLK_{USBC}$.			
15	USBCHFCLKSEL	0	R	**USBC HFCLK Selected**
	HFCLK is selected (and active) as $HFCORECLK_{USBC}$.			
14	CALBSY	0	R	**Calibration Busy**
	Calibration is on-going.			
13	LFXOSEL	0	R	**LFXO Selected**
	LFXO is selected as HFCLK clock source.			
12	LFRCOSEL	0	R	**LFRCO Selected**
	LFRCO is selected as HFCLK clock source.			
11	HFXOSEL	0	R	**HFXO Selected**
	HFXO is selected as HFCLK clock source.			
10	HFRCOSEL	1	R	**HFRCO Selected**
	HFRCO is selected as HFCLK clock source.			
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.			
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.			
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.			

L.5.13 CMU_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	1
Access																									R	R	R	R	R	R	R	R
Name																									USBCHFCLKSEL	CALOF	CALRDY	AUXHFRCORDY	LFXORDY	LFRCORDY	HFXORDY	HFRCORDY

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7	USBCHFCLKSEL	0	R	**USBC HFCLK Selected Interrupt Flag**
	Set when HFCLK is selected as $HFCORECLK_{USBC}$.			
6	CALOF	0	R	**Calibration Overflow Interrupt Flag**
	Set when calibration overflow has occurred			
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).			
2	LFRCORDY	0	R	**LFRCO Ready Interrupt Flag**
	Set when LFRCO is ready (start-up time exceeded).			
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).			

L.5.14 CMU_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
Access																									W1	W1	W1	W1	W1	W1	W1	W1
Name																									USBCHFCLKSEL	CALOF	CALRDY	AUXHFRCORDY	LFXORDY	LFRCORDY	HFXORDY	HFRCORDY

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7	USBCHFCLKSEL	0	W1	**USBC HFCLK Selected Interrupt Flag Set**
	Write to 1 to set the USBC HFCLK Selected Interrupt Flag.			
6	CALOF	0	W1	**Calibration Overflow Interrupt Flag Set**
	Write to 1 to set the Calibration Overflow Interrupt Flag.			
5	CALRDY	0	W1	**Calibration Ready Interrupt Flag Set**
	Write to 1 to set the Calibration Ready(completed) Interrupt Flag.			
4	AUXHFRCORDY	0	W1	**AUXHFRCO Ready Interrupt Flag Set**
	Write to 1 to set the AUXHFRCO Ready Interrupt Flag.			
3	LFXORDY	0	W1	**LFXO Ready Interrupt Flag Set**

Bit	Name	Reset	Access	Description
	Write to 1 to set the LFXO Ready Interrupt Flag.			
2	LFRCORDY	0	W1	**LFRCO Ready Interrupt Flag Set**
	Write to 1 to set the LFRCO Ready Interrupt Flag.			
1	HFXORDY	0	W1	**HFXO Ready Interrupt Flag Set**
	Write to 1 to set the HFXO Ready Interrupt Flag.			
0	HFRCORDY	0	W1	**HFRCO Ready Interrupt Flag Set**
	Write to 1 to set the HFRCO Ready Interrupt Flag.			

L.5.15 CMU_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
Access																									W1	W1	W1	W1	W1	W1	W1	W1
Name																									USBCHFCLKSEL	CALOF	CALRDY	AUXHFRCORDY	LFXORDY	LFRCORDY	HFXORDY	HFRCORDY

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7	USBCHFCLKSEL	0	W1	**USBC HFCLK Selected Interrupt Flag Clear**
	Write to 1 to clear the USBC HFCLK Selected Interrupt Flag.			
6	CALOF	0	W1	**Calibration Overflow Interrupt Flag Clear**
	Write to 1 to clear the Calibration Overflow Interrupt Flag.			
5	CALRDY	0	W1	**Calibration Ready Interrupt Flag Clear**
	Write to 1 to clear the Calibration Ready Interrupt Flag.			
4	AUXHFRCORDY	0	W1	**AUXHFRCO Ready Interrupt Flag Clear**
	Write to 1 to clear the AUXHFRCO Ready Interrupt Flag.			
3	LFXORDY	0	W1	**LFXO Ready Interrupt Flag Clear**
	Write to 1 to clear the LFXO Ready Interrupt Flag.			
2	LFRCORDY	0	W1	**LFRCO Ready Interrupt Flag Clear**
	Write to 1 to clear the LFRCO Ready Interrupt Flag.			
1	HFXORDY	0	W1	**HFXO Ready Interrupt Flag Clear**
	Write to 1 to clear the HFXO Ready Interrupt Flag.			
0	HFRCORDY	0	W1	**HFRCO Ready Interrupt Flag Clear**
	Write to 1 to clear the HFRCO Ready Interrupt Flag.			

L.5.16 CMU_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
Access																									RW	RW	RW	RW	RW	RW	RW	RW
Name																									USBCHFCLKSEL	CALOF	CALRDY	AUXHFRCORDY	LFXORDY	LFRCORDY	HFXORDY	HFRCORDY

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7	USBCHFCLKSEL	0	RW	**USBC HFCLK Selected Interrupt Enable**
	Set to enable the USBC HFCLK Selected Interrupt.			
6	CALOF	0	RW	**Calibration Overflow Interrupt Enable**
	Set to enable the Calibration Overflow Interrupt.			
5	CALRDY	0	RW	**Calibration Ready Interrupt Enable**
	Set to enable the Calibration Ready Interrupt.			
4	AUXHFRCORDY	0	RW	**AUXHFRCO Ready Interrupt Enable**
	Set to enable the AUXHFRCO Ready Interrupt.			
3	LFXORDY	0	RW	**LFXO Ready Interrupt Enable**
	Set to enable the LFXO Ready Interrupt.			
2	LFRCORDY	0	RW	**LFRCO Ready Interrupt Enable**
	Set to enable the LFRCO Ready Interrupt.			
1	HFXORDY	0	RW	**HFXO Ready Interrupt Enable**
	Set to enable the HFXO Ready Interrupt.			
0	HFRCORDY	0	RW	**HFRCO Ready Interrupt Enable**
	Set to enable the HFRCO Ready Interrupt.			

L.5.17 CMU_HFCORECLKEN0 - High Frequency Core Clock Enable Register 0

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

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5	NC	NC	NC	NC
4	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.			
3	USB	0	RW	**Universal Serial Bus Interface Clock Enable**
	Set to enable the clock for USB.			
2	USBC	0	RW	**Universal Serial Bus Interface Core Clock Enable**
	Set to enable the clock for USBC.			
1	AES	0	RW	**Advanced Encryption Standard Accelerator Clock Enable**
	Set to enable the clock for AES.			
0	DMA	0	RW	**Direct Memory Access Controller Clock Enable**
	Set to enable the clock for DMA.			

L.5.18 CMU_HFPERCLKEN0 - High Frequency Peripheral Clock Enable Register 0

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															DAC0	ADC0	PRS	VCMP	GPIO	I2C1	I2C0	ACMP1	ACMP0	TIMER3	TIMER2	TIMER1	TIMER0	UART1	UART0	USART2	USART1	USART0

Bit	Name	Reset	Access	Description
31:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17	DAC0	0	RW	**Digital to Analog Converter 0 Clock Enable**
	Set to enable the clock for DAC0.			
16	ADC0	0	RW	**Analog to Digital Converter 0 Clock Enable**
	Set to enable the clock for ADC0.			
15	PRS	0	RW	**Peripheral Reflex System Clock Enable**
	Set to enable the clock for PRS.			
14	VCMP	0	RW	**Voltage Comparator Clock Enable**
	Set to enable the clock for VCMP.			
13	GPIO	0	RW	**General purpose Input/Output Clock Enable**
	Set to enable the clock for GPIO.			
12	I2C1	0	RW	**I2C 1 Clock Enable**
	Set to enable the clock for I2C1.			
11	I2C0	0	RW	**I2C 0 Clock Enable**
	Set to enable the clock for I2C0.			
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.			
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	UART1	0	RW	**Universal Asynchronous Receiver/Transmitter 1 Clock Enable**
	Set to enable the clock for UART1.			
3	UART0	0	RW	**Universal Asynchronous Receiver/Transmitter 0 Clock Enable**
	Set to enable the clock for UART0.			
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.			

L.5.19 CMU_SYNCBUSY - Synchronization Busy 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
Access																										R		R		R		R
Name																										LFBPRESC0		LFBCLKEN0		LFAPRESC0		LFACLKEN0

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		

Bit	Name	Reset	Access	Description
6	LFBPRESC0	0	R	**Low Frequency B Prescaler 0 Busy**
	Used to check the synchronization status of CMU_LFBPRESC0.			

Value	Description
1	CMU_LFBPRESC0 is busy synchronizing new value.

5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.		
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.		
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.

L.5.20 CMU_FREEZE - Freeze 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
Access																																RW
Name																																REGFREEZE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.

L.5.21 CMU_LFACLKEN0 - Low Frequency A Clock Enable Register 0 (Async Reg)

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
Access																													RW	RW	RW	RW

Offset	Bit Position																													
Name																												LETIMER0	RTC	LESENSE

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	LETIMER0	0	RW	**Low Energy Timer 0 Clock Enable**
	Set to enable the clock for LETIMER0.			
1	RTC	0	RW	**Real-Time Counter Clock Enable**
	Set to enable the clock for RTC.			
0	LESENSE	0	RW	**Low Energy Sensor Interface Clock Enable**
	Set to enable the clock for LESENSE.			

L.5.22 CMU_LFBCLKEN0 - Low Frequency B Clock Enable Register 0 (Async Reg)

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
Access																															RW	RW
Name																															LEUART1	LEUART0

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

L.5.23 CMU_LFAPRESC0 - Low Frequency A Prescaler Register 0 (Async Reg)

Offset	Bit Position																													
0x068	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																			0x0		0x0				0x0						0x0	
Access																			RW		RW				RW						RW	
Name																					LETIMER0				RTC						LESENSE	

Bit	Name	Reset	Access	Description
31:14	Reserved	To ensure compatibility with future devices, always write bits to 0.		
13:12	N/A	N/A	N/A	N/A
11:8	LETIMER0	0x0	RW	**Low Energy Timer 0 Prescaler**
	Configure Low Energy Timer 0 prescaler			

Bit	Name	Reset	Access	Description
	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
7:4	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
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1:0	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

L.5.24 CMU_LFBPRESC0 - Low Frequency B Prescaler Register 0 (Async Reg)

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

Offset	Bit Position																											
Name																										LEUART1		LEUART0

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.		
1:0	LEUART0	0x0	RW	**Low Energy UART 0 Prescaler**
	Configure Low Energy UART 0 prescaler			

Value	Mode	Description
0	DIV1	$LFBCLK_{LEUART0}$ = LFBCLK
1	DIV2	$LFBCLK_{LEUART0}$ = LFBCLK/2
2	DIV4	$LFBCLK_{LEUART0}$ = LFBCLK/4
3	DIV8	$LFBCLK_{LEUART0}$ = LFBCLK/8

L.5.25 CMU_PCNTCTRL - PCNT Control 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	0	0	0	0	0
Access																											RW	RW	RW	RW	RW	RW
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.		
5	PCNT2CLKSEL	0	RW	**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.

4	PCNT2CLKEN	0	RW	**PCNT2 Clock Enable**
	This bit enables/disables the clock to the PCNT.			

Value	Description
0	PCNT2 is disabled.
1	PCNT2 is enabled.

3	PCNT1CLKSEL	0	RW	**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	RW	**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	RW	**PCNT0 Clock Select**
	This bit controls which clock that is used for the PCNT.			

Value	Mode	Description
0	LFACLK	LFACLK is clocking PCNT0.
1	PCNT0S0	External pin PCNT0_S0 is clocking PCNT0.

Bit	Name	Reset	Access	Description
0	PCNT0CLKEN	0	RW	**PCNT0 Clock Enable**
	This bit enables/disables the clock to the PCNT.			

Value	Description
0	PCNT0 is disabled.
1	PCNT0 is enabled.

L.5.26 CMU_ROUTE - I/O Routing 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			0	0
Access																												RW			RW	RW
Name																												LOCATION			CLKOUT1PEN	CLKOUT0PEN

Bit	Name	Reset	Access	Description
31:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4:2	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the CMU I/O pins.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

Bit	Name	Reset	Access	Description
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.			

L.5.27 CMU_LOCK - Configuration Lock 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																	0x0000															
Access																	RW															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	LOCKKEY	0x0000	RW	**Configuration Lock Key**

Write any other value than the unlock code to lock CMU_CTRL, CMU_HFCORECLKDIV, CMU_HFPERCLKDIV, CMU_HFRCOCTRL, CMU_LFRCOCTRL, CMU_AUXHFRCOCTRL, CMU_OSCENCMD, CMU_CMD, CMU_LFCLKSEL, CMU_HFCORECLKEN0, CMU_HFPERCLKEN0, CMU_LFACLKEN0, CMU_LFBCLKEN0, CMU_LFAPRESC0, CMU_LFBPRESC0, 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.

M ARM Watchdog Timer

M.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 may e.g. be caused by an external event, such as an ESD pulse, or by a software failure.

M.2 Features

- Clock input from selectable oscillators
  - Internal 32.768 Hz RC oscillator
  - Internal 1 kHz RC oscillator
  - External 32.768 Hz XTAL oscillator
- Configurable timeout period from 9 to 256k watchdog clock cycles
- Individual selection to keep running or freeze when entering EM2 or EM3
- 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

M.3 Functional Description

The watchdog is enabled by setting the EN bit in WDOG_CTRL. When enabled, the watchdog counts up to the period value configured through the PERSEL field in WDOG_CTRL. If the watchdog timer is not cleared to 0 (by writing a 1 to the CLEAR bit in WDOG_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 WDOG_CTRL. Once locked, it cannot be disabled or reconfigured by software.

The watchdog counter is reset when EN is reset.

M.3.1 Clock Source

Three clock sources are available for use with the watchdog, through the CLKSEL field in WDOG_CTRL. The corresponding clocks must be enabled in the CMU. The SWOSCBLOCK bit in WDOG_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 WDOG_CTRL is used to divide the selected watchdog clock, and the timeout for the watchdog timer can be calculated like this:

$$T_{TIMEOUT} = (2^{3+PERSEL} + 1)/f \quad (3)$$

where f is the frequency of the selected clock.

It is recommended to clear the watchdog first, if PERSEL is changed while the watchdog is enabled.

To use this module, the LE interface clock must be enabled in CMU_HFCORECLKEN0, in addition to the module clock.

M.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 WDOG_CTRL. When code execution is resumed, the watchdog will continue counting where it left off.

M.3.3 Energy Mode Handling

The watchdog timer can be configured to either keep on running or freeze when entering EM2 or EM3. The configuration is done individually for each energy mode in the EM2RUN and EM3RUN bits in WDOG_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 and EM1. The watchdog does not run in EM4, and if EM4BLOCK in WDOG_CTRL is set, the CPU is prevented from entering EM4.

If the WDOG is clocked by the LFXO or LFRCO, writing the SWOSCBLOCK bit will effectively prevent the CPU from entering EM3. When running from the ULFRCO, writing the SWOSCBLOCK bit will prevent the CPU from entering EM4.

M.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. Please refer to [?] for a description on how to perform register accesses to Low Energy Peripherals. note that clearing the EN bit in WDOG_CTRL will reset the WDOG module, which will halt any ongoing register synchronization.

Never write to the WDOG registers when it is disabled, except to enable it by setting the EN bitfield in WDOG_CTRL. Make sure that the enable is registered (i.e. WDOG_SYNCBUSY_CTRL goes low), before writing other registers.

M.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

M.5 Register Description

M.5.1 WDOG_CTRL - Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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		0xF					0	0	0	0	0	0	0
Access																			RW		RW					RW	RW	RW	RW	RW	RW	RW
Name																			CLKSEL		PERSEL					SWOSCBLOCK	EM4BLOCK	LOCK	EM3RUN	EM2RUN	DEBUGRUN	EN

Bit	Name	Reset	Access	Description
31:14	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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

11:8	PERSEL	0xF	RW	**Watchdog Timeout Period Select**
	Select watchdog timeout period.			

Bit	Name	Reset	Access	Description

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.
15	Timeout period of 256k watchdog clock cycles.

Bit	Name	Reset	Access	Description
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.

Bit	Name	Reset	Access	Description
5	EM4BLOCK	0	RW	**Energy Mode 4 Block**

Set to prevent the EMU from entering EM4.

Value	Description
0	EM4 can be entered. See EMU for detailed description.
1	EM4 cannot be entered.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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.

M.5.2 WDOG_CMD - Command Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
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.

M.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
Access																															R	R
Name																															CMD	CTRL

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

N ARM Peripheral Reflex System

N.1 Introduction

The Peripheral Reflex System (PRS) system is a network which allows the different peripheral modules to communicate directly with each other without involving the CPU. Peripheral modules which send out reflex signals are called producers. The PRS routes these reflex signals to consumer peripherals which apply 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.

N.2 Features

- 12 configurable interconnect channels
  - Each channel can be connected to any producing peripheral
  - Consumers can choose which channel to listen to
  - Selectable edge detector (rising, falling and both edges)
- Software controlled channel output
  - Configurable level
  - Triggered pulses

N.3 Functional Description

An overview of the PRS module is shown in Figure 88. The PRS contains 12 interconnect channels, and each of these can select between all the output reflex signals offered by the producers. The consumers can then 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 pulse signals and level signals. Synchronous PRS pulses are one HFPERCLK cycle long, and can either be sent out by a producer (e.g., ADC conversion complete) or be generated from the edge detector in the PRS channel. Level signals can have an arbitrary waveform (e.g., Timer PWM output).

N.3.1 Asynchronous Mode

Many reflex signals can operate in two modes, synchronous or asynchronous. A synchronous reflex is clocked on HFPERCLK, and can be used as an input to all reflex consumers, but since they require HFPERCLK, they will not work in EM2/EM3.

Asynchronous reflexes are not clocked on HFPERCLK, and can be used even in EM2/EM3. There is a limitation to reflexes operating in asynchronous mode though: they can only be used by a subset of the reflex consumers, the ones marked with async support in Table 90. Peripherals that can produce asynchronous reflexes are marked with async support in Table 89. To use these reflexes asynchronously, set ASYNC in the CHCTRL register for the PRS channel selecting the reflex signal.

If a peripheral channel with ASYNC set is used in a consumer not supporting asynchronous reflexes, the behaviour is undefined.

N.3.2 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. It is also possible to generate output reflex signals by configuring the SWPULSE and SWLEVEL bits. SWLEVEL is a programmable level for each channel and holds the value it is programmed to. The SWPULSE will give out a one-cycle high pulse if it is written to 1, otherwise a 0 is asserted. The SWLEVEL and SWPULSE 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.

The edge detector controlled by EDSEL should only be used when working with synchronous reflexes, i.e., ASYNC in CHCTRL is cleared.

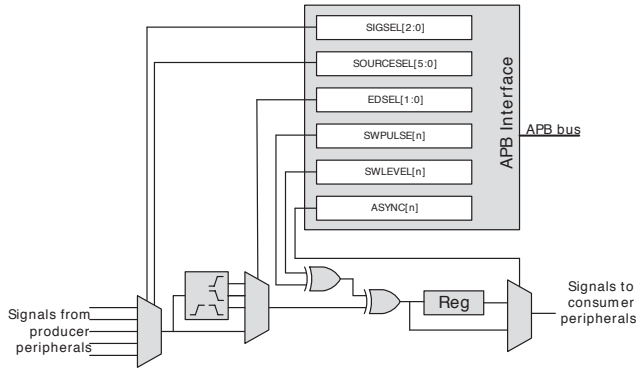


Figure 88: PRS Overview

N.3.3 Producers

Each PRS channel can choose between signals from several producers, which is configured in SOURCESEL in PRS_CHx_CTRL. Each of these producers 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 is given in Table 89.

N.3.4 Consumers

Consumer peripherals (listed in Table 90) can be set to listen to a PRS channel and perform an action based on the signal received on that channel. Most consumers expect pulse input, while some can handle level inputs as well.

It is possible to output prs channel 0 - channel 3 onto the GPIO by setting CH0PEN,

Module	Reflex Output	Output Format	Async Support
ACMP	Comparator Output	Level	Yes
ADC	Single Conversion Done	Pulse	
	Scan Conversion Done	Pulse	
DAC	Channel 0 Conversion Done	Pulse	
	Channel 1 Conversion Done	Pulse	
GPIO	Pin 0 Input	Level	Yes
	Pin 1 Input	Level	Yes
	Pin 2 Input	Level	Yes
	Pin 3 Input	Level	Yes
	Pin 4 Input	Level	Yes
	Pin 5 Input	Level	Yes
	Pin 6 Input	Level	Yes
	Pin 7 Input	Level	Yes
	Pin 8 Input	Level	Yes
	Pin 9 Input	Level	Yes
	Pin 10 Input	Level	Yes
	Pin 11 Input	Level	Yes
	Pin 12 Input	Level	Yes
	Pin 13 Input	Level	Yes
	Pin 14 Input	Level	Yes
	Pin 15 Input	Level	Yes
RTC	Overflow	Pulse	Yes
	Compare Match 0	Pulse	Yes
	Compare Match 1	Pulse	Yes
TIMER	Underflow	Pulse	
	Overflow	Pulse	
	CC0 Output	Level	
	CC1 Output	Level	
	CC2 Output	Level	
LETIMER	CH0	Level	Yes
	CH1	Level	Yes
UART	TX Complete	Pulse	
	RX Data Received	Pulse	
USART	TX Complete	Pulse	
	RX Data Received	Pulse	
	IrDA Decoder Output	Level	
VCMP	Comparator Output	Level	Yes
LESENSE	SCANRES register	Level	Yes
	Decoder Output	Level/Pulse	Yes
BURTC	Overflow	Pulse	Yes
	Compare match 0	Pulse	Yes

Figure 89: Reflex Producers

Module	Reflex Input	Input Format	Async Support
ADC	Single Mode Trigger	Pulse	
	Scan Mode Trigger	Pulse	
DAC	Channel 0 Trigger	Pulse	
	Channel 1 Trigger	Pulse	
TIMER	CC0 Input	Pulse/Level	
	CC1 Input	Pulse/Level	
	CC2 Input	Pulse/Level	
	DTI Fault Source 0 (TIMER0 only)	Pulse	
	DTI Fault Source 1 (TIMER0 only)	Pulse	
	DTI Input (TIMER0 only)	Pulse/Level	
UART	TX/RX Enable	Pulse	
	RX Input	Pulse/Level	Yes
USART	TX/RX Enable	Pulse	
	IrDA Encoder Input (USART0 only)	Pulse	
	RX Input	Pulse/Level	Yes
LEUART	RX Input	Pulse/Level	Yes
PCNT	S0 input	Level	Yes
	S1 input	Level	Yes
LESENSE	Start scan	Pulse/Level	Yes
	Decoder Bit 0	Level	Yes
	Decoder Bit 1	Level	Yes
	Decoder Bit 2	Level	Yes
	Decoder Bit 3	Level	Yes

Figure 90: Reflex Consumers

CH1PEN, CH2PEN, or CH3PEN in the PRS_ROUTE register.

N.3.5 Example

The example below (illustrated in Figure 91) 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 0b011100 to select TIMER0 as input to PRS channel 5.
- Set SIGSEL in PRS_CH5_CTRL to 0b001 to select the overflow signal (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.

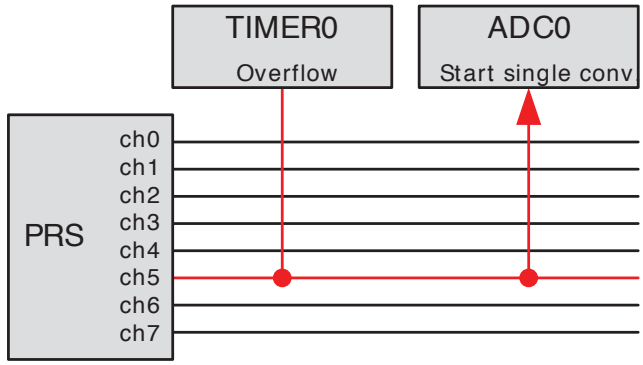


Figure 91: TIMER0 overflow starting ADC0 single conversions through PRS channel 5.

N.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_ROUTE	RW	I/O Routing Register
0x010	PRS_CH0_CTRL	RW	Channel Control Register
...	PRS_CHx_CTRL	RW	Channel Control Register
0x03C	PRS_CH11_CTRL	RW	Channel Control Register

N.5 Register Description

N.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
Access																					W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1
Name																					CH11PULSE	CH10PULSE	CH9PULSE	CH8PULSE	CH7PULSE	CH6PULSE	CH5PULSE	CH4PULSE	CH3PULSE	CH2PULSE	CH1PULSE	CH0PULSE

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			
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**

Bit	Name	Reset	Access	Description
	See bit 0.			
0	CH0PULSE	0	W1	**Channel 0 Pulse Generation**
	Write to 1 to generate one HFPERCLK 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.			

N.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
Access																					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																					CH11LEVEL	CH10LEVEL	CH9LEVEL	CH8LEVEL	CH7LEVEL	CH6LEVEL	CH5LEVEL	CH4LEVEL	CH3LEVEL	CH2LEVEL	CH1LEVEL	CH0LEVEL

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			
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.			

N.5.3 PRS_ROUTE - I/O Routing 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
Access																						RW							RW	RW	RW	RW

Offset	Bit Position																															
Name																LOCATION									CH3PEN	CH2PEN	CH1PEN	CH0PEN					

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the PRS I/O pins.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1

Bit	Name	Reset	Access	Description
7:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	CH3PEN	0	RW	**CH3 Pin Enable**
	When set, GPIO output from PRS channel 3 is enabled			
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			

N.5.4 PRS_CHx_CTRL - 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			0x0				0x00																			0x0		
Access				RW			RW				RW																			RW		
Name				ASYNC			EDSEL				SOURCESEL																			SIGSEL		

Bit	Name	Reset	Access	Description
31:29	Reserved	To ensure compatibility with future devices, always write bits to 0.		
28	ASYNC	0	RW	**Asynchronous reflex**
	Set to disable synchronization of this reflex signal			
27:26	Reserved	To ensure compatibility with future devices, always write bits to 0.		
25:24	EDSEL	0x0	RW	**Edge Detect Select**
	Select edge detection.			

Value	Mode	Description
0	OFF	Signal is left as it is
1	POSEDGE	A one HFPERCLK cycle pulse is generated for every positive edge of the incoming signal
2	NEGEDGE	A one HFPERCLK clock cycle pulse is generated for every negative edge of the incoming signal
3	BOTHEDGES	A one HFPERCLK clock cycle pulse is generated for every edge of the incoming signal

Bit	Name	Reset	Access	Description
23:22	Reserved	To ensure compatibility with future devices, always write bits to 0.		
21:16	SOURCESEL	0x00	RW	**Source Select**
	Select input source to PRS channel.			

Bit	Name	Reset	Access	Description
	Value	**Mode**		**Description**
	0b000000	NONE		No source selected
	0b000001	VCMP		Voltage Comparator
	0b000010	ACMP0		Analog Comparator 0
	0b000011	ACMP1		Analog Comparator 1
	0b000110	DAC0		Digital to Analog Converter 0
	0b001000	ADC0		Analog to Digital Converter 0
	0b010000	USART0		Universal Synchronous/Asynchronous Receiver/Transmitter 0
	0b010001	USART1		Universal Synchronous/Asynchronous Receiver/Transmitter 1
	0b010010	USART2		Universal Synchronous/Asynchronous Receiver/Transmitter 2
	0b011100	TIMER0		Timer 0
	0b011101	TIMER1		Timer 1
	0b011110	TIMER2		Timer 2
	0b011111	TIMER3		Timer 3
	0b100100	USB		Universal Serial Bus Interface
	0b101000	RTC		Real-Time Counter
	0b101001	UART0		Universal Asynchronous Receiver/Transmitter 0
	0b101010	UART1		Universal Asynchronous Receiver/Transmitter 1
	0b110000	GPIOL		General purpose Input/Output
	0b110001	GPIOH		General purpose Input/Output
	0b110100	LETIMER0		Low Energy Timer 0
	0b110111	BURTC		Backup RTC
	0b111001	LESENSEL		Low Energy Sensor Interface
	0b111010	LESENSEH		Low Energy Sensor Interface
	0b111011	LESENSED		Low Energy Sensor Interface
15:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	SIGSEL	0x0	RW	**Signal Select**
	Select signal input to PRS channel.			

Bit	Name	Reset	Access	Description
	Value	Mode		Description
	SOURCESEL = 0b000000 (NONE)			
	0bxxx	OFF		Channel input selection is turned off
	SOURCESEL = 0b000001 (VCMP)			
	0b000	VCMPOUT		Voltage comparator output VCMPOUT
	SOURCESEL = 0b000010 (ACMP0)			
	0b000	ACMP0OUT		Analog comparator output ACMP0OUT
	SOURCESEL = 0b000011 (ACMP1)			
	0b000	ACMP1OUT		Analog comparator output ACMP1OUT
	SOURCESEL = 0b000110 (DAC0)			
	0b000	DAC0CH0		DAC ch0 conversion done DAC0CH0
	0b001	DAC0CH1		DAC ch1 conversion done DAC0CH1
	SOURCESEL = 0b001000 (ADC0)			
	0b000	ADC0SINGLE		ADC single conversion done ADC0SINGLE
	0b001	ADC0SCAN		ADC scan conversion done ADC0SCAN
	SOURCESEL = 0b010000 (USART0)			
	0b000	USART0IRTX		USART 0 IRDA out USART0IRTX
	0b001	USART0TXC		USART 0 TX complete USART0TXC
	0b010	USART0RXDATAV		USART 0 RX Data Valid USART0RXDATAV
	SOURCESEL = 0b010001 (USART1)			
	0b001	USART1TXC		USART 1 TX complete USART1TXC
	0b010	USART1RXDATAV		USART 1 RX Data Valid USART1RXDATAV
	SOURCESEL = 0b010010 (USART2)			
	0b001	USART2TXC		USART 2 TX complete USART2TXC
	0b010	USART2RXDATAV		USART 2 RX Data Valid USART2RXDATAV
	SOURCESEL = 0b011100 (TIMER0)			
	0b000	TIMER0UF		Timer 0 Underflow TIMER0UF
	0b001	TIMER0OF		Timer 0 Overflow TIMER0OF
	0b010	TIMER0CC0		Timer 0 Compare/Capture 0 TIMER0CC0
	0b011	TIMER0CC1		Timer 0 Compare/Capture 1 TIMER0CC1
	0b100	TIMER0CC2		Timer 0 Compare/Capture 2 TIMER0CC2
	SOURCESEL = 0b011101 (TIMER1)			
	0b000	TIMER1UF		Timer 1 Underflow TIMER1UF
	0b001	TIMER1OF		Timer 1 Overflow TIMER1OF
	0b010	TIMER1CC0		Timer 1 Compare/Capture 0 TIMER1CC0
	0b011	TIMER1CC1		Timer 1 Compare/Capture 1 TIMER1CC1
	0b100	TIMER1CC2		Timer 1 Compare/Capture 2 TIMER1CC2
	SOURCESEL = 0b011110 (TIMER2)			
	0b000	TIMER2UF		Timer 2 Underflow TIMER2UF
	0b001	TIMER2OF		Timer 2 Overflow TIMER2OF
	0b010	TIMER2CC0		Timer 2 Compare/Capture 0 TIMER2CC0
	0b011	TIMER2CC1		Timer 2 Compare/Capture 1 TIMER2CC1
	0b100	TIMER2CC2		Timer 2 Compare/Capture 2 TIMER2CC2
	SOURCESEL = 0b011111 (TIMER3)			
	0b000	TIMER3UF		Timer 3 Underflow TIMER3UF
	0b001	TIMER3OF		Timer 3 Overflow TIMER3OF
	0b010	TIMER3CC0		Timer 3 Compare/Capture 0 TIMER3CC0
	0b011	TIMER3CC1		Timer 3 Compare/Capture 1 TIMER3CC1
	0b100	TIMER3CC2		Timer 3 Compare/Capture 2 TIMER3CC2
	SOURCESEL = 0b100100 (USB)			
	0b000	USBSOF		USB Start of Frame USBSOF
	0b001	USBSOFSR		USB Start of Frame Sent/Received USBSOFSR
	SOURCESEL = 0b101000 (RTC)			
X005109,	0b000	RTCOF		RTC Overflow RTCOF
	0b001	RTCCOMP0		RTC Compare 0 RTCCOMP0
	0b010	RTCCOMP1		RTC Compare 1 RTCCOMP1
	SOURCESEL = 0b101001 (UART0)			
	0b001	UART0TXC		USART 0 TX complete UART0TXC
	0b010	UART0RXDATAV		USART 0 RX Data Valid UART0RXDATAV

Bit	Name	Reset	Access	Description

O ARM External Bus Interface

O.1 Introduction

The External Bus Interface provides access to external parallel interface devices such as SRAM, FLASH and ADCs. The interface is memory mapped into the address bus of the Cortex-M3. 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.

O.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-M3 or DMA

O.3 Functional Description

An overview of the EBI module is shown in Figure 92. The EBI module implements a generic external device interface to for example SRAM or Flash devices.

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-M3 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.

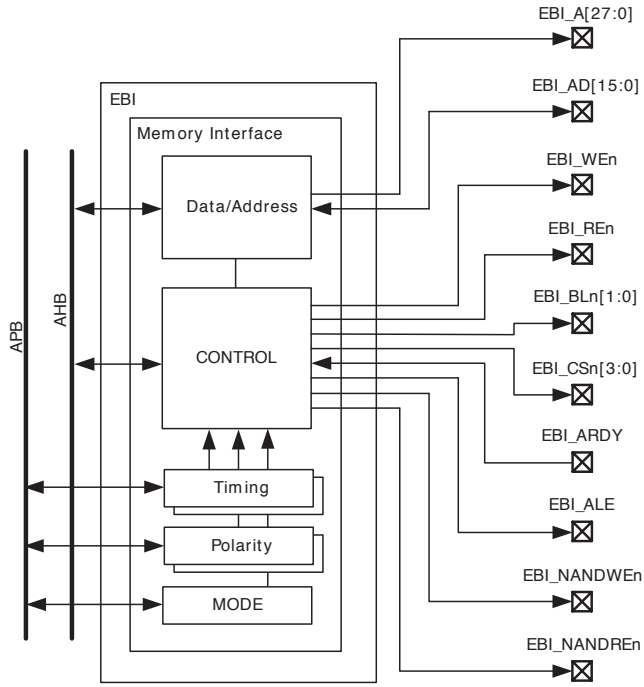


Figure 92: EBI Overview

O.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 Section O.3.6. Read and write signals in 8-bit mode are shown in Figure 93 and Figure 94 respectively.

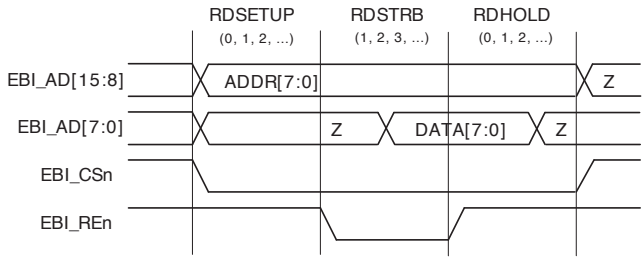


Figure 93: EBI Non-multiplexed 8-bit Data, 8-bit Address Read Operation

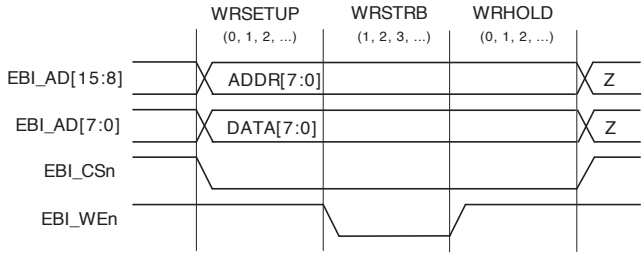


Figure 94: EBI Non-multiplexed 8-bit Data, 8-bit Address Write Operation

O.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 95. This mode is set by programming the MODE field in the EBI_CTRL register to D16A16ALE.

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.

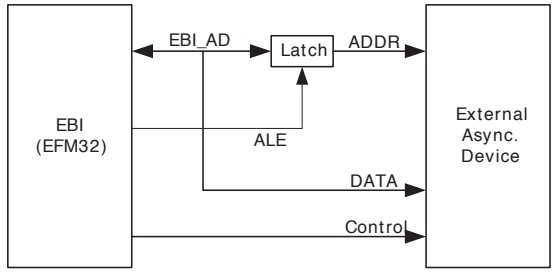


Figure 95: EBI Address Latch Setup

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 96 and Figure 97 respectively.

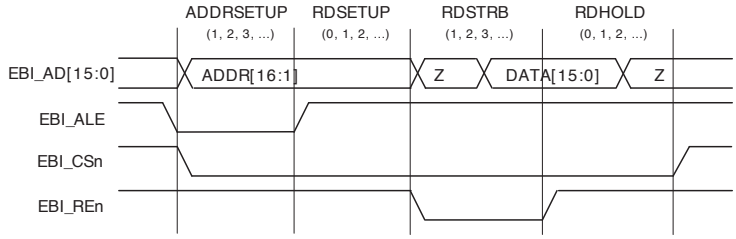


Figure 96: EBI Multiplexed 16-bit Data, 16-bit Address Read Operation

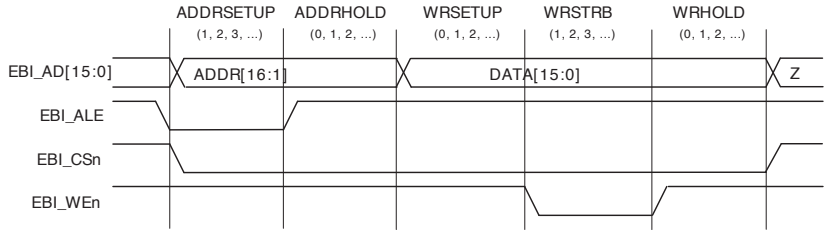


Figure 97: EBI Multiplexed 16-bit Data, 16-bit Address Write Operation

O.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 98 and Figure 99 respectively.

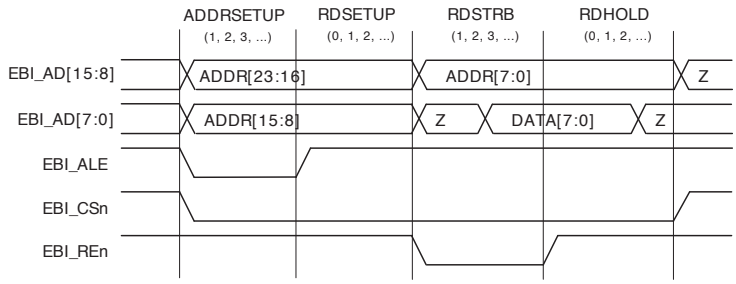


Figure 98: EBI Multiplexed 8-bit Data, 24-bit Address Read Operation

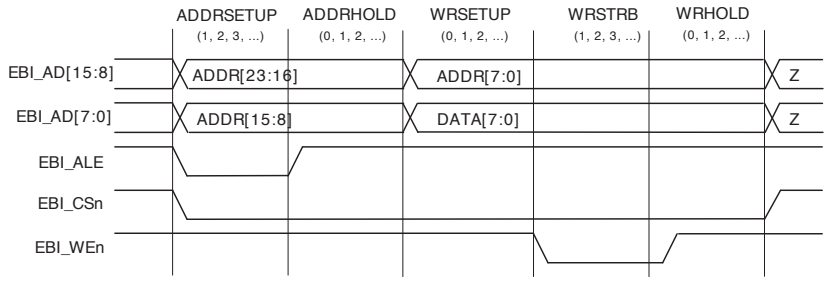


Figure 99: EBI Multiplexed 8-bit Data, 24-bit Address Write Operation

O.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 Section O.3.6. This mode is set by programming the MODE field in the EBI_CTRL register to D16. Read and write signals are shown in Figure 100 and Figure 101 respectively for the case in which N address lines on EBI_A have been enabled.

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.

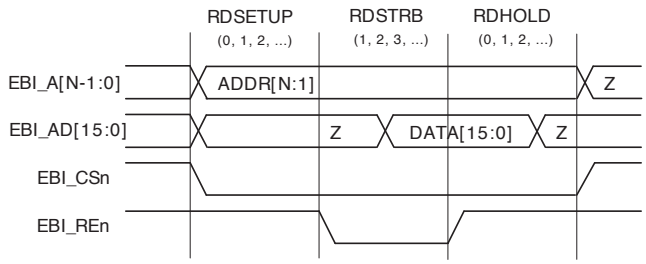


Figure 100: EBI Non-multiplexed 16-bit Data Read Operation with Extended Address

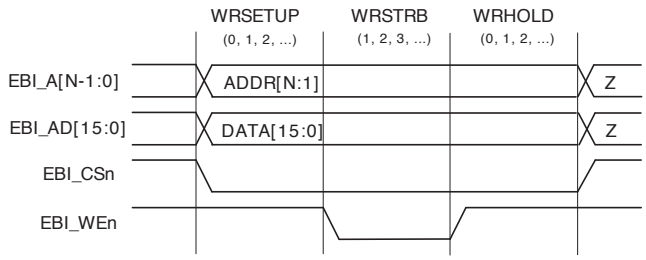


Figure 101: EBI Non-multiplexed 16-bit Data Write Operation with Extended Address

O.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 102.

Figure 102: 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 93, Figure 96, Figure 98 or Figure 100 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 103: EBI Page Mode Read Operation for D8A8 addressing mode

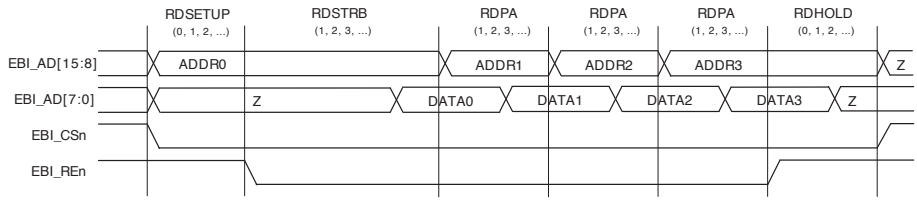


Figure 104: EBI Page Mode Read Operation for D16A16ALE addressing mode

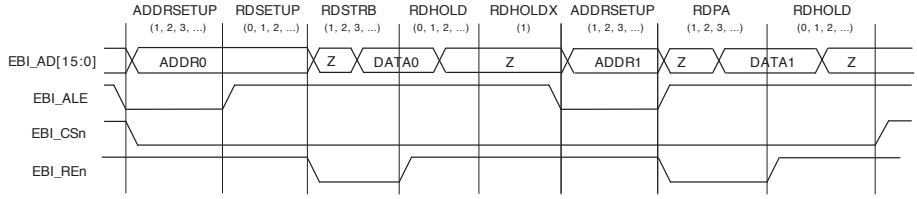


Figure 105: EBI Page Mode Read Operation for D8A24ALE addressing mode

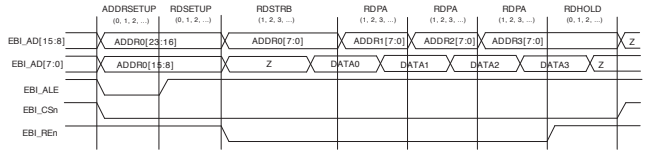
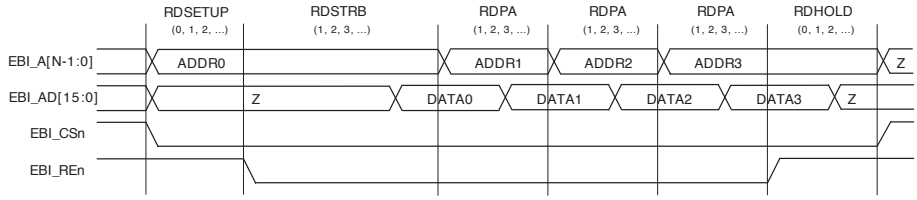


Figure 106: EBI Page Mode Read Operation for D16 addressing mode



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 107 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.

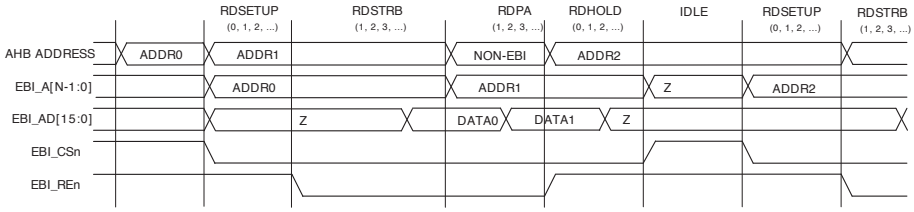


Figure 107: EBI Page Closing

O.3.6 Extended addressing

Extended addressing is used to extend the address range for any of the addressing modes described in Section O.3.4, Section O.3.1, Section O.3.2 and Section O.3.3. 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 Section O.3.12 for the memory map definitions related to the EBI. An example of address extension for the D16 mode is shown in Figure 100 and Figure 101. A further example for address extension in the multiplexed 16-bit data, 16-bit address mode of Section O.3.2 is shown in Figure 108. 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_ROUTE register.

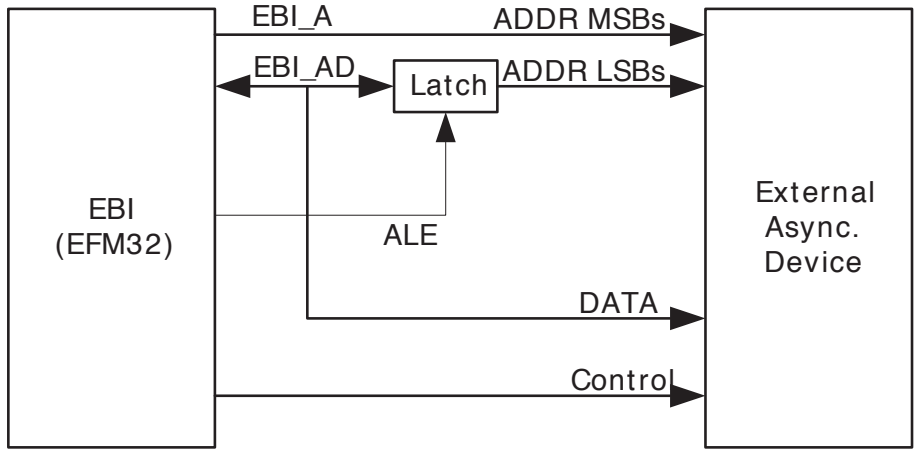


Figure 108: EBI Extended Address Latch Setup

Read and write signals for using extended addressing in the D16A16ALE mode are shown in Figure 109 and Figure 110 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.

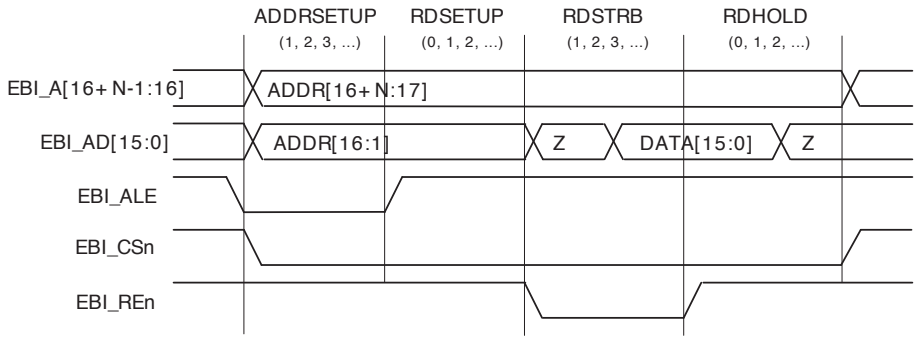


Figure 109: EBI 16-bit Data Multiplexed Read Operation using Extended Addressing

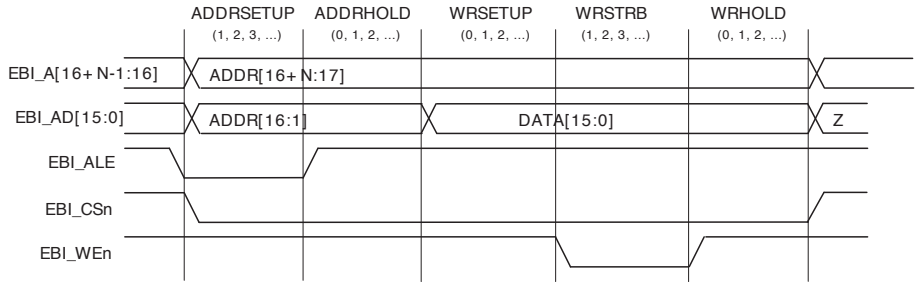


Figure 110: EBI 16-bit Data Multiplexed Write Operation using Extended Addressing

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_ROUTE 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_ROUTE and by enabling the required higher address lines via the APEN bitfield in EBI_ROUTE. The operation of the APEN and ALB bitfields is shown in Table 111 for some typical configurations.

Figure 111: 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]

O.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-M3 or DMA read transaction handled by the EBI. Note that one prefetch transaction might lead to multiple external device transactions as described in Table 116. 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-M3 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.

O.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 96 and Figure 97 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.

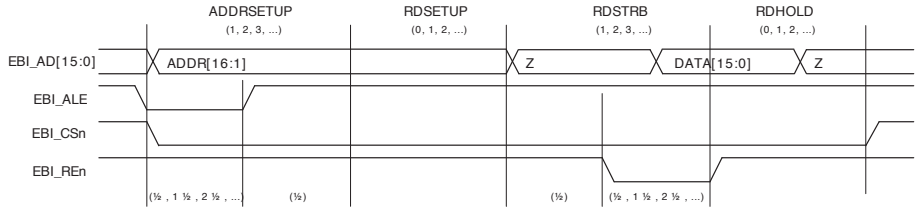


Figure 112: EBI Multiplexed Read Operation with Reduced Length Strobes

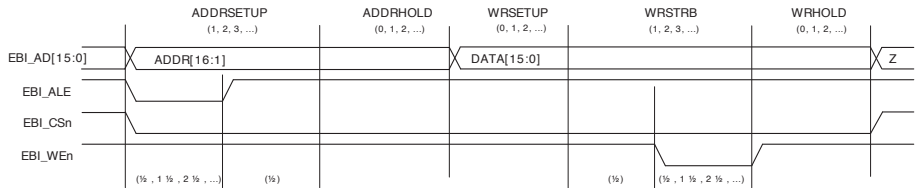


Figure 113: EBI Multiplexed Write Operation with Reduced Length Strobes

O.3.9 Bus turn-around and Idle cycles

The EBI_AD lines can be driven by either the ARM core 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 114. 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 115 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.

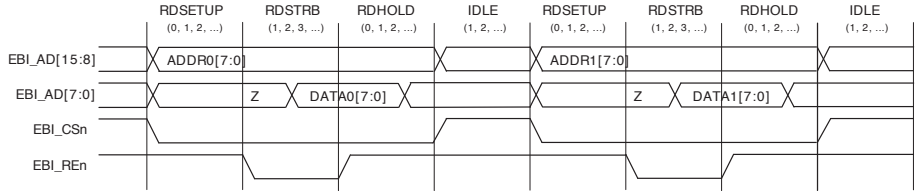


Figure 114: EBI Enforced IDLE cycles between Transactions

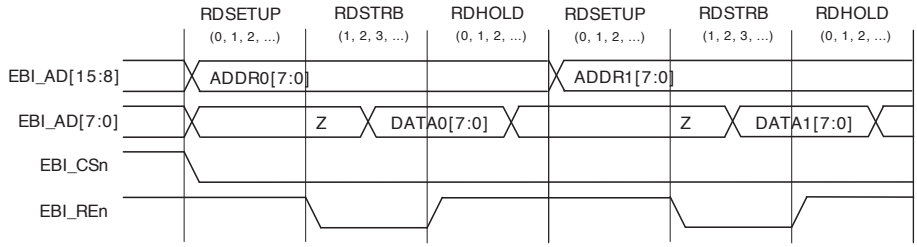


Figure 115: EBI No Enforced IDLE cycles between Transactions

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.

O.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 99 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.

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.

O.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 116:

- 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.

Figure 116: EBI Mapping of AHB Transactions to External Device Transactions

Data Access by Cortex-M3, DMA, or prefetch	8-bit External Device (non-NAND) transaction(s)	16-bit External Device (non-NAND) transac-tion(s)(with byte lanes)	16-bit External Device (non-NAND) transac-tion(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

O.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 117 and Figure 118 respectively.

The EBI regions starting at address 0x80000000 in the memory map of the ARM core can also be used for code execution. When running code via EBI regions starting at this address, the Cortex-M3 uses the System bus interface to fetch instructions. This results in reduced performance as the Cortex-M3 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-M3 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.

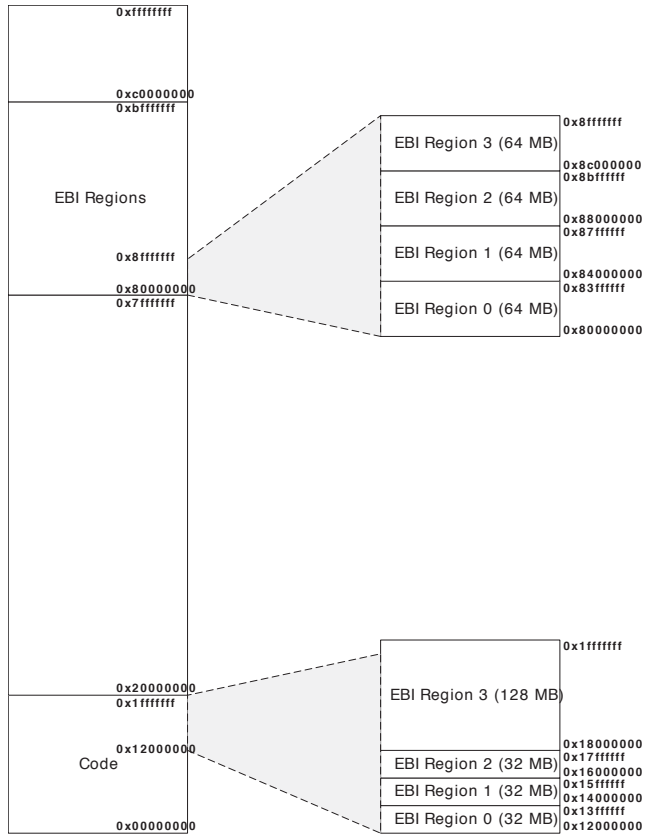


Figure 117: EBI Default Memory Map (ALTMAP = 0)

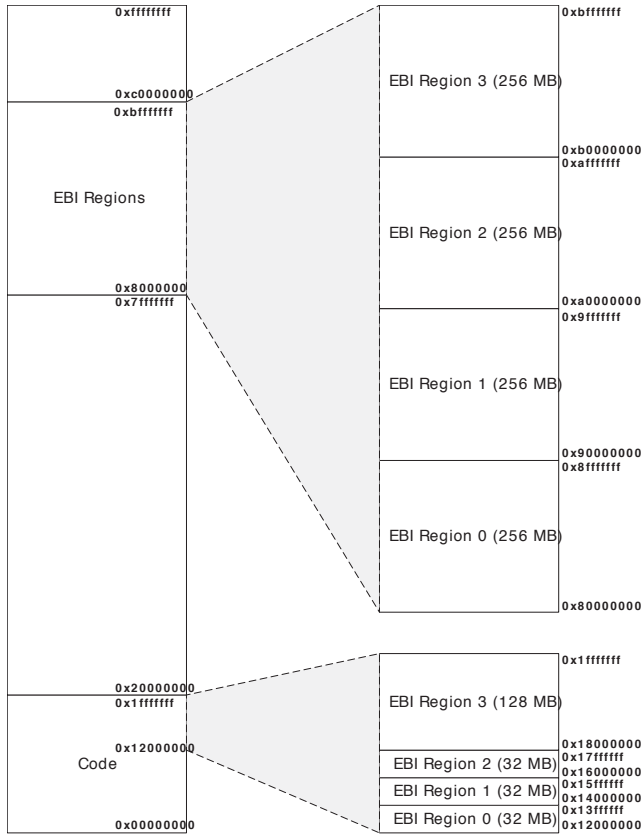


Figure 118: EBI Alternative Memory Map (ALTMAP = 1)

O.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 ARDYEN and ARDYTODIS only apply to bank 0. Wait behavior for bank n is then defined in the ARDYnEN and ARDYTOnDIS bitfields.

O.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 Section O.3.14. ECC support is described in Section O.3.15.

Both standard and Chip Enable Don't Care (CEDC) NAND Flash devices are supported and they can be attached as shown in Figure 119 and Figure 120 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.

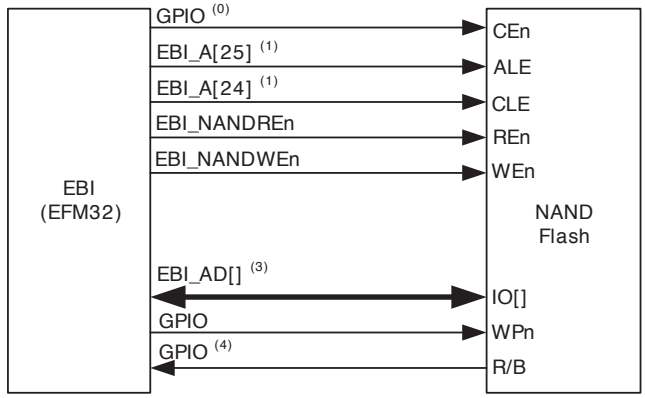


Figure 119: EBI Connection with Standard NAND Flash

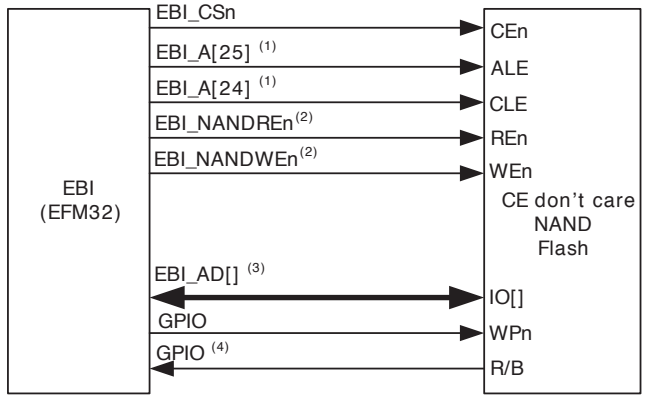


Figure 120: EBI Connection with Chip Enable Don't Care NAND Flash

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 Section O.3.14
- (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)

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 ARM core. 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 121. 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 119 and Figure 120 are just an example.

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

ALE	CLE	Selected NAND Flash Register
0	0	Data Register
0	1	Command Register
1	0	Address Register
1	1	Undefined

Figure 121: EBI NAND Flash Register Select

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 Section O.3.11. 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 122, Figure 123, and Figure 124 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.

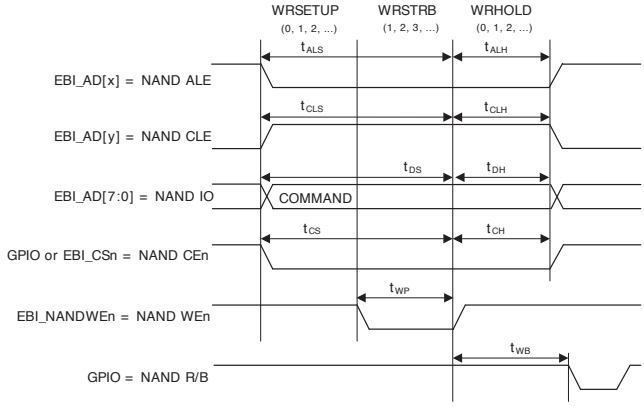


Figure 122: EBI NAND Flash Command Latch Timing

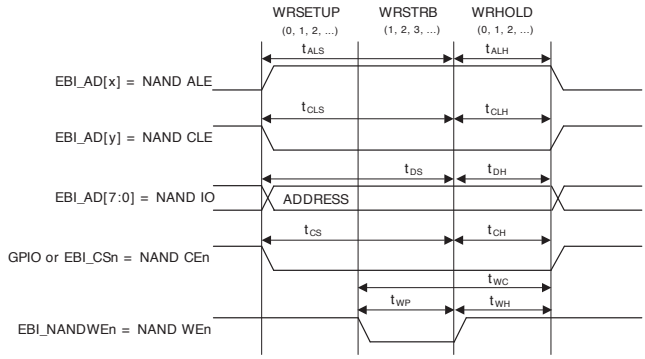


Figure 123: EBI NAND Flash Address Latch Timing

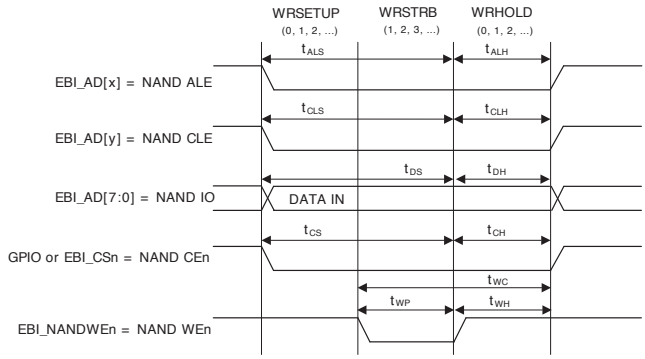


Figure 124: EBI NAND Flash Data Input Timing

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 125.

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)
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)

Figure 125: EBI NAND Flash Write Timing

NAND Flash read timing is defined in the EBI_RDTIMING(n) register. Figure 126 shows the NAND Flash data output timing assuming the D8A8 address mode.

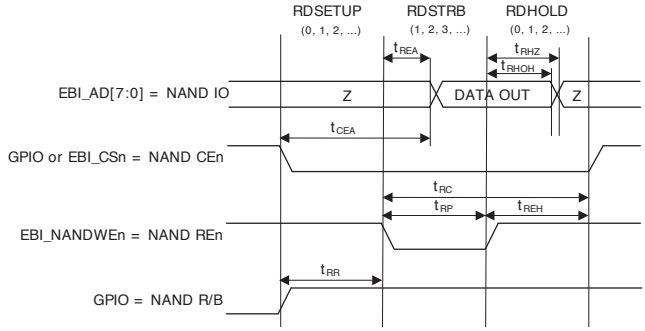


Figure 126: EBI NAND Flash Data Output Timing

The EBI_RDTIMING(n) setting requirements for satisfying the NAND Flash timing parameters for data output timing are shown in Table 127.

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)
tCLR	<= t(RDSETUP)
tIR	<= t(RDSETUP)

Figure 127: EBI NAND Flash Read Timing

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 128.

NAND Timing Parameter	EBI Timing Parameter
tWHR	<= t(WRHOLD) + t(RDSETUP)
tRHW	<= t(RDHOLD) + t(WRSETUP)

Figure 128: EBI NAND Flash Read/Write Timing Requirements

Remaining NAND Flash timing parameters, e.g. tRST and tPROG, should be dealt with in software.

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-M3 or DMA writes to the memory mapped NAND Command and Address registers. The automatic data access width conversions described in Section O.3.11 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.

O.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 129 and Table 130. 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.

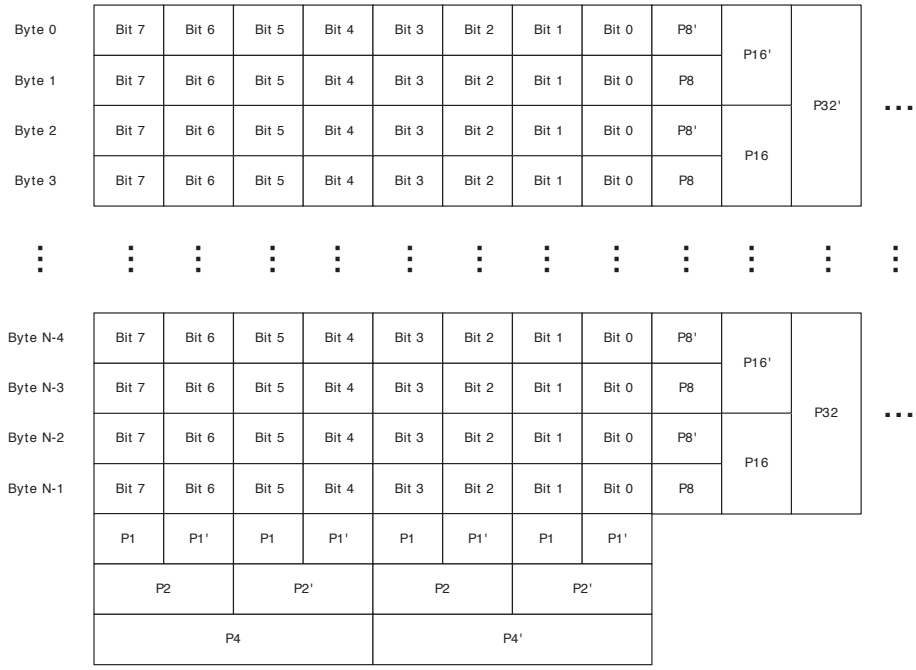


Figure 129: EBI ECC Generation

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'
P4	Bit 7 xor Bit 6 xor Bit 5 xor Bit 4 xor P4

Figure 130: EBI ECC Bit/Column 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.

Figure 131: EBI ECC Byte/Row Parity

The generated ECC code can be read from the EBI_ECCPARITY register according to the format shown in Figure 132. 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 133.

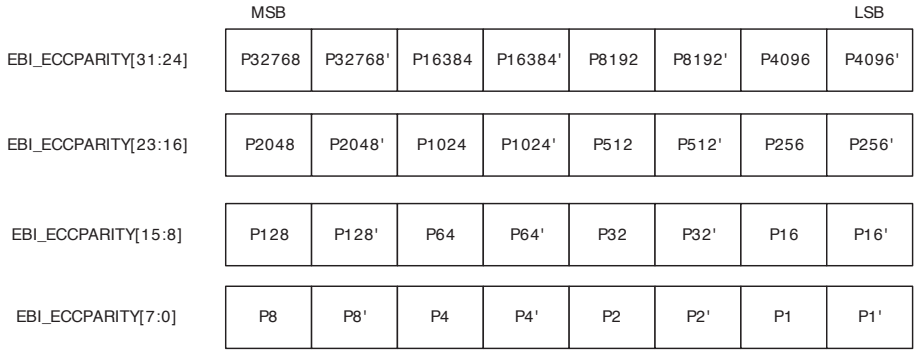


Figure 132: EBI EBI_ECCPARITY Format

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]
2048	EBI_ECCPARITY[27:0]
4096	EBI_ECCPARITY[29:0]
8192	EBI_ECCPARITY[31:0]

Figure 133: EBI EBI_ECCPARITY valid bits

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 134.

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

Figure 134: EBI Error Detection Result

O.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
0x014	EBI_ROUTE	RW	I/O Routing Register

Offset	Name	Type	Description
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
0x098	EBI_IF	R	Interrupt Flag Register
0x09C	EBI_IFS	W1	Interrupt Flag Set Register
0x0A0	EBI_IFC	W1	Interrupt Flag Clear Register
0x0A4	EBI_IEN	RW	Interrupt Enable Register

O.5 Register Description

O.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**

Bit	Name	Reset	Access	Description
	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 [?]		
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.			
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 on EBI_A in the EBI_ROUTE register.
1	D16A16ALE	EBI_AD drives 16 bit data, 16 bit address, ALE is used for address latching. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.
2	D8A24ALE	EBI_AD drives 8 bit data, 24 bit address, ALE is used for address latching. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.
3	D16	EBI_AD drives 16 bit data, ALE not used. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.

5:4	MODE2	0x0	RW	**Mode 2**

Bit	Name	Reset	Access	Description

This field sets the access mode the EBI will use for interfacing devices on bank 2. 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 on EBI_A in the EBI_ROUTE register.
1	D16A16ALE	EBI_AD drives 16 bit data, 16 bit address, ALE is used for address latching. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.
2	D8A24ALE	EBI_AD drives 8 bit data, 24 bit address, ALE is used for address latching. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.
3	D16	EBI_AD drives 16 bit data, ALE not used. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.

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 on EBI_A in the EBI_ROUTE register.
1	D16A16ALE	EBI_AD drives 16 bit data, 16 bit address, ALE is used for address latching. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.
2	D8A24ALE	EBI_AD drives 8 bit data, 24 bit address, ALE is used for address latching. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.
3	D16	EBI_AD drives 16 bit data, ALE not used. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.

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 on EBI_A in the EBI_ROUTE register.
1	D16A16ALE	EBI_AD drives 16 bit data, 16 bit address, ALE is used for address latching. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.
2	D8A24ALE	EBI_AD drives 8 bit data, 24 bit address, ALE is used for address latching. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.
3	D16	EBI_AD drives 16 bit data, ALE not used. Extended address bits can be enabled on EBI_A in the EBI_ROUTE register.

O.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																			0x3								0x3	
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 [?]		
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.			
27:10	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
9:8	ADDRHOLD	0x3	RW	**Address Hold Time**
	Sets the number of cycles the address is held after ALE is asserted.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	ADDRSETUP	0x3	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.			

O.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											0x3				0x3F												0x3	
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 [?]		
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.			
27:18	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
17:16	RDHOLD	0x3	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.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:8	RDSTRB	0x3F	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.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	RDSETUP	0x3	RW	**Read Setup Time**
	Sets the number of cycles the address setup before REn is asserted.			

O.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											0x3				0x3F												0x3	
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 [?]		
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.			
27:18	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
17:16	WRHOLD	0x3	RW	**Write Hold Time**
	Sets the number of cycles CSn is held active after the WEn is deasserted.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:8	WRSTRB	0x3F	RW	**Write Strobe Time**
	Sets the number of cycles the WEn is held active. If set to 0, 1 cycle is inserted by HW.			

Bit	Name	Reset	Access	Description
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	WRSETUP	0x3	RW	**Write Setup Time**
	Sets the number of cycles the address setup before WEn is asserted.			

O.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 [?]		
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.

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.

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.

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.

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.
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.

O.5.6 EBI_ROUTE - I/O Routing 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				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	RW
Name		LOCATION				CSTFTPEN	DATAENPEN	TFTPEN		APEN					ALB					NANDPEN					BLPEN	ARDYPEN	ALEPEN	CS3PEN	CS2PEN	CS1PEN	CS0PEN	EBIPEN

Bit	Name	Reset	Access	Description
31	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
30:28	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the EBI I/O pins.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

27	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
26	CSTFTPEN	0	RW	**EBI_CSTFT Pin Enable**
	When set, the EBI_CSTFT pin is enabled			
25	DATAENPEN	0	RW	**EBI_TFT 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 [?]		
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.
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.			

Bit	Name	Reset	Access	Description
	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 [?]		
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 [?]		
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			
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			

O.5.7 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																			0x3								0x3	
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 [?]		
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.			
27:10	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
9:8	ADDRHOLD	0x3	RW	**Address Hold Time**
	Sets the number of cycles the address is held after ALE is asserted.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	ADDRSETUP	0x3	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.			

O.5.8 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											0x3				0x3F												0x3	
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 [?]		
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.			
27:18	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
17:16	RDHOLD	0x3	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.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:8	RDSTRB	0x3F	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.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	RDSETUP	0x3	RW	**Read Setup Time**
	Sets the number of cycles the address setup before REn is asserted.			

O.5.9 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											0x3				0x3F												0x3	
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 [?]		
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.			
27:18	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
17:16	WRHOLD	0x3	RW	**Write Hold Time**
	Sets the number of cycles CSn is held active after the WEn is deasserted.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:8	WRSTRB	0x3F	RW	**Write Strobe Time**
	Sets the number of cycles the WEn is held active. If set to 0, 1 cycle is inserted by HW.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	WRSETUP	0x3	RW	**Write Setup Time**
	Sets the number of cycles the address setup before WEn is asserted.			

O.5.10 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 [?]		
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.
1	ACTIVEHIGH	REn and NANDREn are active high.

Bit	Name	Reset	Access	Description
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.

O.5.11 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																			0x3							0x3	
Access				RW																			RW							RW	

Offset	Bit Position																													
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 [?]		
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.			
27:10	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
9:8	ADDRHOLD	0x3	RW	**Address Hold Time**
	Sets the number of cycles the address is held after ALE is asserted.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	ADDRSETUP	0x3	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.			

O.5.12 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											0x3				0x3F												0x3	
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 [?]		
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.			
27:18	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
17:16	RDHOLD	0x3	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.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:8	RDSTRB	0x3F	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.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	RDSETUP	0x3	RW	**Read Setup Time**
	Sets the number of cycles the address setup before REn is asserted.			

O.5.13 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											0x3				0x3F												0x3	

Offset	Bit Position																											
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 [?]		
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.			
27:18	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
17:16	WRHOLD	0x3	RW	**Write Hold Time**
	Sets the number of cycles CSn is held active after the WEn is deasserted.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:8	WRSTRB	0x3F	RW	**Write Strobe Time**
	Sets the number of cycles the WEn is held active. If set to 0, 1 cycle is inserted by HW.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	WRSETUP	0x3	RW	**Write Setup Time**
	Sets the number of cycles the address setup before WEn is asserted.			

O.5.14 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 [?]		
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.

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.

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.

2	WEPOL	0	RW	**Write Enable Polarity**
	Sets the polarity of the EBI_WEn and EBI_NANDWEn lines.			

Bit	Name	Reset	Access	Description
	Value	**Mode**		**Description**
	0	ACTIVELOW		WEn and NANDWEn are active low.
	1	ACTIVEHIGH		WEn and NANDWEn are active high.
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.
	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.

O.5.15 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																			0x3								0x3	
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 [?]		
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.			
27:10	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
9:8	ADDRHOLD	0x3	RW	**Address Hold Time**
	Sets the number of cycles the address is held after ALE is asserted.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	ADDRSETUP	0x3	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.			

O.5.16 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											0x3				0x3F												0x3	
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 [?]		

Bit	Name	Reset	Access	Description
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.			
27:18	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
17:16	RDHOLD	0x3	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.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:8	RDSTRB	0x3F	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.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	RDSETUP	0x3	RW	**Read Setup Time**
	Sets the number of cycles the address setup before REn is asserted.			

O.5.17 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											0x3				0x3F												0x3	
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 [?]		
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.			
27:18	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
17:16	WRHOLD	0x3	RW	**Write Hold Time**
	Sets the number of cycles CSn is held active after the WEn is deasserted.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:8	WRSTRB	0x3F	RW	**Write Strobe Time**
	Sets the number of cycles the WEn is held active. If set to 0, 1 cycle is inserted by HW.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
1:0	WRSETUP	0x3	RW	**Write Setup Time**
	Sets the number of cycles the address setup before WEn is asserted.			

O.5.18 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

Offset	Bit Position						
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 [?]		
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.

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.

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.

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.

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.
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.

O.5.19 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																0x7						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 [?]		
26:20	KEEPOPEN	0x00	RW	**Maximum Page Open Time.**

Bit	Name	Reset	Access	Description
	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.			
19:11	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
10:8	RDPA	0x7	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 [?]		
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 [?]		
1:0	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.

O.5.20 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 [?]		
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 [?]		
0	EN	0	RW	**NAND Flash control enable**
	This field enables NAND Flash control for the memory bank defined in BANK.			

O.5.21 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 [?]		
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.			

O.5.22 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 [?]		
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 [?]		
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 [?]		
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 [?]		
0	AHBACT	0	R	**EBI Busy with AHB Transaction.**
	Indicates that EBI is busy with an AHB Transaction.			

O.5.23 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.			

O.5.24 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								0			0x0					0				0	0x0		0	0				0x0			0x0	
Access								RW			RW					RW				RW	RW		RW	RW				RW			RW	
Name								RGBMODE			BANKSEL					WIDTH				COLOR1SRC	INTERLEAVE		FBCTRIG	SHIFTDCLKEN				MASKBLEND			DD	

Bit	Name	Reset	Access	Description
31:25	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
24	RGBMODE	0	RW	**TFT RGB Mode**
	This field sets TFT RGB Mode.			

Value	Mode	Description
0	RGB565	RGB data is 565.
1	RGB555	RGB data is 555.

Bit	Name	Reset	Access	Description
23:22	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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.

Bit	Name	Reset	Access	Description
19:17	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
16	WIDTH	0	RW	**TFT Transaction Width**
	This field sets TFT tranaction width.			

Value	Mode	Description
0	BYTE	TFT Data is 8 bit wide.
1	HALFWORD	TFT Data is 16 bit wide.

Bit	Name	Reset	Access	Description
15:13	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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.			

Bit	Name	Reset	Access	Description

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.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
7:5	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
4:2	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	IMASKIALPHA	Internal Masking and Alpha Blending are enabled.
5	EMASK	External Masking is enabled.
6	EALPHA	External Alpha Blending is enabled.
7	EMASKEALPHA	External Masking and Alpha Blending are enabled.

Bit	Name	Reset	Access	Description
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.

O.5.25 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						0x000																0x000										
Access						R																R										
Name						VCNT																HCNT										

Bit	Name	Reset	Access	Description
31:27	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
26:16	VCNT	0x000	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 [?]		
10:0	HCNT	0x000	R	**Horizontal Count**
	Contains the current pixel position within a line (initial pixel in horizontal backporch has HCNT = 0).			

O.5.26 EBI_TFTFRAMEBASE - TFT Frame Base 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					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 [?]		
27:0	FRAMEBASE	0x0000000	RW	**Frame Base Address**
	Sets the frame base address.			

O.5.27 EBI_TFTSTRIDE - TFT Stride 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	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 [?]		
11:0	HSTRIDE	0x000	RW	**Horizontal Stride**
	Sets the horizontal stride added to the Direct Drive address at the end of each line.			

O.5.28 EBI_TFTSIZE - TFT Size 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	0x000																															
Access	RW																															
Name	VSZ																															

Bit	Name	Reset	Access	Description
31:26	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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 [?]		
9:0	HSZ	0x000	RW	**Horizontal Size (excluding porches)**
	Sets the horizontal size in pixels. Set to required size minus 1.			

O.5.29 EBI_TFTHPORCH - TFT Horizontal Porch 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				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 [?]		
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 [?]		
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 [?]		
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 [?]		
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.			

O.5.30 EBI_TFTVPORCH - TFT Vertical Porch 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						
Access							RW										RW									RW						
Name							VBPORCH										VFPORCH									VSYNC						

Bit	Name	Reset	Access	Description
31:26	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
25:18	VBPORCH	0x00	RW	**Vertical Back Porch Size**
	Sets the Vertical back porch size in pixels.			
17:16	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
15:8	VFPORCH	0x00	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 [?]		
6:0	VSYNC	0x00	RW	**Vertical Synchronization Pulse Width**
	Sets the Vertical synchronization pulse width. Set to required width minus 1.			

O.5.31 EBI_TFTTIMING - TFT Timing 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				0x0			0x000												0x000									
Access			RW				RW			RW												RW									
Name			TFTHOLD				TFTSETUP			TFTSTART												DCLKPERIOD									

Bit	Name	Reset	Access	Description
31:30	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
29:28	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.			
27:26	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		

Bit	Name	Reset	Access	Description
25:24	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	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
22:12	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	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
10:0	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.			

O.5.32 EBI_TFTPOLARITY - TFT Polarity 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
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 [?]		
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.

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.

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.

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.

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.
1	ACTIVEHIGH	CSTFT is active high.

O.5.33 EBI_TFTDD - TFT Direct Drive Data 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	0x0000																															
Access	RW																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
15:0	DATA	0x0000	RW	**TFT Direct Drive Data from Internal Memory**
	Sets the RGB value used when Direct Drive from internal memory is used (DD = INTERNAL)			

O.5.34 EBI_TFTALPHA - TFT Alpha Blending 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	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 [?]		
8:0	ALPHA	0x000	RW	**TFT Alpha Blending Factor**
	Sets the alpha blending factor. The maximum value is 256.			

O.5.35 EBI_TFTPIXEL0 - TFT Pixel 0 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	0x0000																															
Access	RW																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
15:0	DATA	0x0000	RW	**RGB data.**
	Sets the RGB data value according to the format defined in RGBMODE.			

O.5.36 EBI_TFTPIXEL1 - TFT Pixel 1 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	0x0000																															
Access	RW																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
15:0	DATA	0x0000	RW	**RGB data.**
	Sets the RGB data value according to the format defined in RGBMODE.			

O.5.37 EBI_TFTPIXEL - TFT Alpha Blending Result Pixel 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	0x0000																															
Access	R																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
15:0	DATA	0x0000	R	**Alpha Blending Result**
	RGB result of Alpha Blending operation according to the format defined in RGBMODE.			

O.5.38 EBI_TFTMASK - TFT Masking 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	0x0000																															
Access	RW																															
Name	TFTMASK																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
15:0	TFTMASK	0x0000	RW	**TFT Mask Value**
	Sets the mask value. Data write transactions matching this value are suppressed.			

O.5.39 EBI_IF - Interrupt Flag 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	0	0	0	0	0	0	0	0	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	R	R	R	R	R	R	R
Name	DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC																										

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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**

Bit	Name	Reset	Access	Description
	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.			

O.5.40 EBI_IFS - Interrupt Flag Set 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																											0	0	0	0	0	0
Access																											W1	W1	W1	W1	W1	W1
Name																											DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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.			

O.5.41 EBI_IFC - Interrupt Flag Clear 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
Access																											W1	W1	W1	W1	W1	W1
Name																											DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
5	DDJIT	0	W1	**Direct Drive Jitter Interrupt Flag Clear**

Bit	Name	Reset	Access	Description
	Write to 1 to clear Direct Drive Jitter Interrupt flag.			
4	DDEMPTY	0	W1	**Direct Drive Data Empty Interrupt Flag Clear**
	Write to 1 to clear Direct Drive Data Empty Interrupt flag.			
3	VFPORCH	0	W1	**Vertical Front Porch Interrupt Flag Clear**
	Write to 1 to clear Vertical Front Porch interrupt flag.			
2	VBPORCH	0	W1	**Vertical Back Porch Interrupt Flag Clear**
	Write to 1 to clear Vertical Back Porch interrupt flag.			
1	HSYNC	0	W1	**Horizontal Sync Interrupt Flag Clear**
	Write to 1 to clear Horizontal Sync interrupt flag.			
0	VSYNC	0	W1	**Vertical Sync Interrupt Flag Clear**
	Write to 1 to clear Vertical Sync interrupt flag.			

O.5.42 EBI_IEN - Interrupt Enable 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
Access																											RW	RW	RW	RW	RW	RW
Name																											DDJIT	DDEMPTY	VFPORCH	VBPORCH	HSYNC	VSYNC

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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.			

P ARM InterIntegrated Circuit Interface

P.1 Introduction

The I^2C module provides an interface between the MCU and a serial I^2C-bus. It is capable of acting as both master and 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 I^2C module allows both fine-grained control of the transmission process and close to automatic transfers. Automatic recognition of slave addresses is provided in all energy modes (except EM4).

P.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
- Double buffered data
- Full DMA support

P.3 Functional Description

An overview of the I^2C module is shown in Figure 135.

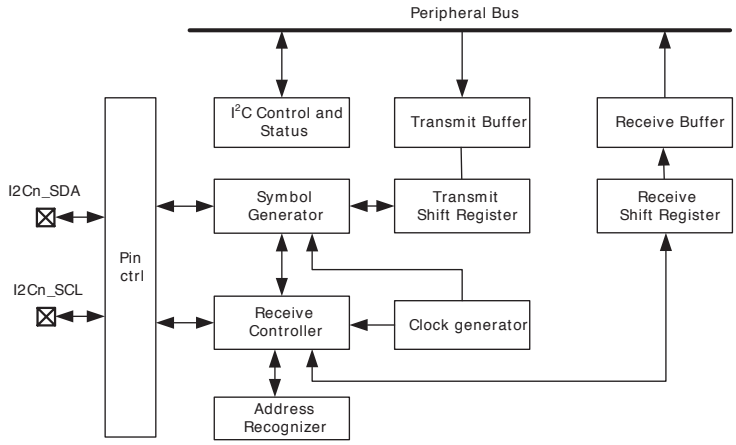


Figure 135: I²C Overview

P.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 136. 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.

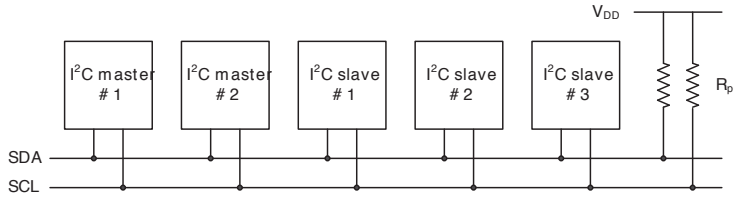


Figure 136: I²C-Bus Example

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 Equation 4.

$$Rp(max) = (tr/0.8473) \times Cb \quad (4)$$

The maximal rise times for 100 kHz, 400 kHz and 1 MHz I²C are 1 μs, 300 ns and 120 ns respectively.

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.

START and STOP Conditions

START and STOP conditions are used to initiate and stop transactions on the I^{2}C-bus. All transactions on the bus begin with a START condition (S) and end with a STOP condition (P). As shown in Figure 137, 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.

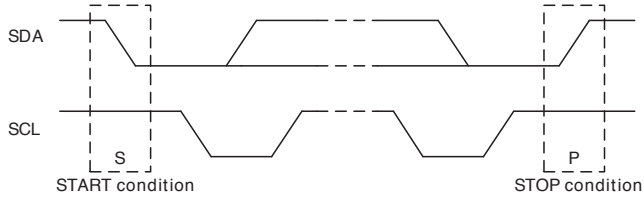


Figure 137: I^{2}C START and STOP Conditions

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 I^{2}C-bus as shown in Figure 136.

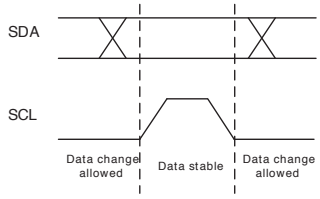


Figure 138: I^{2}C Bit Transfer on I^{2}C-Bus

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^{2}C transfers are shown in Figure 139, Figure 140, and Figure 141. 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 139: I^{2}C Single Byte Write to Slave

Figure 140: I^{2}C Double Byte Read from Slave

Figure 141: I^{2}C Single Byte Write, then Repeated Start and Single Byte Read

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 142, and include a General Call address which can be used to broadcast a message to all slaves on the I²C-bus.

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

Figure 142: I²C Reserved I²C Addresses

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 eight 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 143.

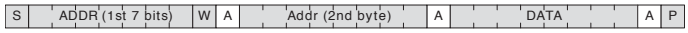


Figure 143: I²C Master Transmitter/Slave Receiver with 10-bit Address

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 144.

Figure 144: I^2C Master Receiver/Slave Transmitter with 10-bit Address

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.

P.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.

When re-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.

P.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.

P.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 Equation P.3.4:

$$f_{SCL} = 1/(T_{low} + T_{high}) \quad (5)$$

where T_{low} and T_{high} is the low and high periods of the clock signal respectively, given below. When the clock is not stretched, the low and high periods of the clock signal are:

$$T_{high} = (N_{high} \times (CLKDIV + 1))/f_{HFPERCLK}, T_{low} = (N_{low} \times (CLKDIV + 1))/f_{HFPERCLK} \quad (6)$$

Equation P.3.4 and Equation P.3.4 does not apply for low clock division factors (0, 1 and 2) because of synchronization. For these clock division factors, the formulas for computing high and low periods of the clock signal are given in Table 145.

Figure 145: I^2C High and Low Periods for Low CLKDIV

CLKDIV	Standard (4:4)		Asymmetric (6:3)		Fast (11:6)	
	T_{low}	T_{high}	T_{low}	T_{high}	T_{low}	T_{high}
0	$7/f_{HFPERCLK}$	$7/f_{HFPERCLK}$	$9/f_{HFPERCLK}$	$6/f_{HFPERCLK}$	$14/f_{HFPERCLK}$	$9/f_{HFPERCLK}$
1	$10/f_{HFPERCLK}$	$10/f_{HFPERCLK}$	$14/f_{HFPERCLK}$	$8/f_{HFPERCLK}$	$24/f_{HFPERCLK}$	$14/f_{HFPERCLK}$
2	$15/f_{HFPERCLK}$	$15/f_{HFPERCLK}$	$21/f_{HFPERCLK}$	$12/f_{HFPERCLK}$	$36/f_{HFPERCLK}$	$21/f_{HFPERCLK}$

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. The available modes

are summarized in Table 146along with the highest I^{2}C-bus frequencies in the given modes that can be achieved without violating the timing specifications of the I^{2}C-bus. The frequencies are calculated taking the maximum allowed rise and fall times of SDA and SCL into account. Higher frequencies may be achieved in practice. The 3 extra cycles are synchronization, and must be taken into consideration when DIV in the I2Cn_CLKDIV register has a low value. The maximum data hold time is dependent on the DIV and is given by:

$$t_{HD,DAT\text{-}max} = (4 + DIV)/f_{HFPERCLK} \quad (7)$$

DIV must be set to 1 during slave mode operation.

Figure 146: I^{2}C Clock Mode

HFPERCLK frequency (MHz)	Clock Low High Ratio (CLHR)	Sm max frequency (kHz)	Fm max frequency (kHz)	Fm+ max frequency (kHz)
48	0	92	400	1000
	1	82	400	1000
	2	72	400	842
28	0	92	400	1000
	1	81	400	848
	2	71	400	736
21	0	90	400	1000
	1	80	400	954
	2	72	368	552
14	0	92	400	1000
	1	81	400	636
	2	68	368	608
11	0	91	400	785
	1	81	333	733
	2	71	289	478
6.6	0	91	400	471
	1	81	299	439
	2	64	286	286
1.2	0	59	85	85
	1	54	79	79
	2	52	52	52

P.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 I^2C module attempts to change its value. If the sensed value is different than the value the I^2C module tried to output, it is interpreted as a simultaneous transmission by another device, and that the I^2C module has lost arbitration.

Whenever arbitration is lost, the ARBLOST interrupt flag in I2Cn_IF is set, any lines held are released, and the I^2C device goes idle. If an I^2C master loses arbitration during the transmission of an address, another master may be trying to address it. The master therefore receives the rest of the address, and if the address matches the slave address of the master, the master goes into either slave transmitter or slave receiver mode.

Arbitration can be lost both when operating as a master and when operating as a slave.

P.3.6 Buffers

Transmit Buffer and Shift Register

The I^2C transmitter is double buffered through the transmit buffer and transmit shift register as shown in Figure 135. A byte is loaded into the transmit buffer by writing to I2Cn_TXDATA. 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.

Whenever a byte is loaded from the transmit buffer to the transmit shift register, the TXBL flag in I2Cn_STATUS and the TXBL interrupt flag in I2Cn_IF are set. This indicates that there is room in the buffer for more data. TXBL is cleared automatically when data is written to the buffer.

If a write is attempted to the transmit buffer while it is not empty, 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 I^2C 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.

Receive Buffer and Shift Register

Like the transmitter, the I^{2}C receiver is double buffered. The receiver uses the receive buffer and receive shift register as shown in Figure 135. 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. Otherwise, the byte waits in the shift register until space becomes available in the buffer.

When a byte becomes available in the receive buffer, the RXDATAV in I2Cn_STATUS and RXDATAV interrupt flag in I2Cn_IF are set. The data can now be fetched from the buffer using I2Cn_RXDATA. Reading from this register will pull a byte out of the buffer, making room for a new byte and clearing RXDATAV in I2Cn_STATUS and RXDATAV in I2Cn_IF in the process.

If a read from the receive buffer is attempted through I2Cn_RXDATA while the buffer is empty, the RXUF interrupt flag in I2Cn_IF is set, and the data read from the buffer is undefined.

I2Cn_RXDATAP 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, but the data read through I2Cn_RXDATAP when the receive buffer is empty is still undefined.

P.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^{2}C module generate a start condition whenever the bus becomes free.

The I^{2}C-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^{2}C-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.

Master State Machine

The master state machine is shown in Figure 147. 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.

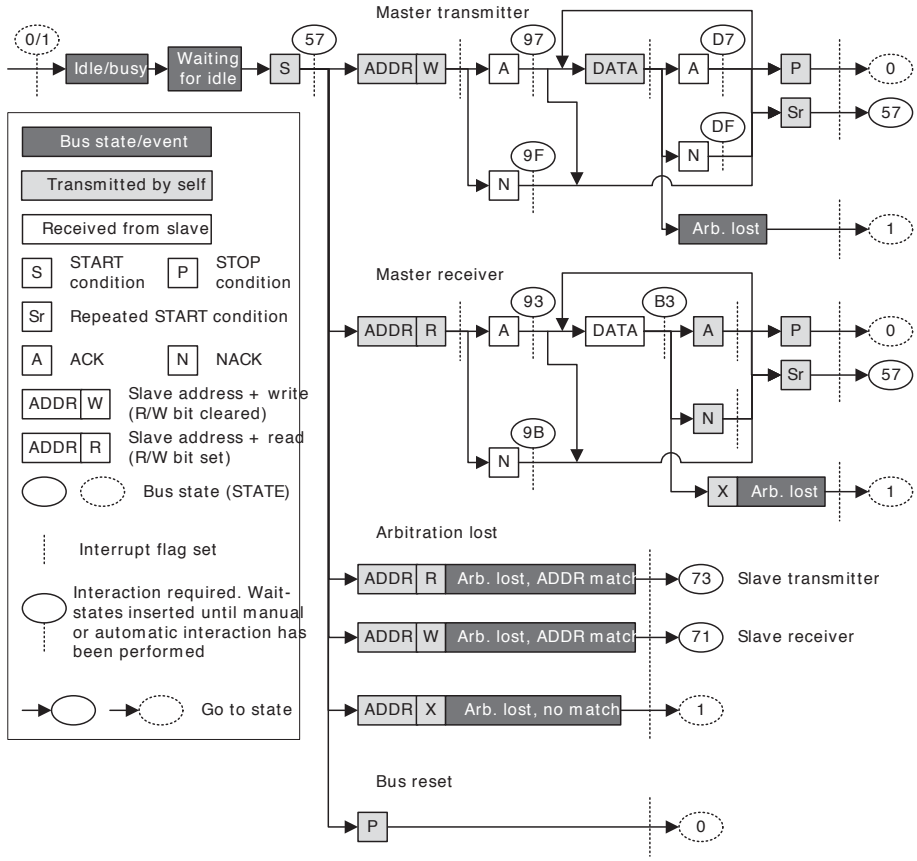


Figure 147: I²C Master State Machine

Interactions

Whenever the I^{2}C 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 I^{2}C module. This state can be read from the I2Cn_STATE register.

As an example, Table 149 shows the different states the I^{2}C 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 I^{2}C module is waiting for.

The bus would never stop at state 0x57 if the address was available in the transmit buffer.

The different interactions used by the I^{2}C module are listed in Table 148 in prioritized order. If a set of different courses of action are possible from a given state, the course of action using the highest priority interactions, that first has everything it is waiting for is the one that is taken.

The commands marked with a * in Table 148 can be issued before an interaction is required. When such a command is issued before it can be used/consumed by the I^{2}C 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 I^{2}C 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 148 is applied to the bus.

Pending commands can be cleared by setting the CLEARPC command bit in I2Cn_CMD.

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 can be used to reduce the amount of software interaction required during a transfer.

Reset State

After a reset, the state of the I^{2}C-bus is unknown. To avoid interrupting transfers on the I^{2}C-bus after a reset of the I^{2}C module or the entire MCU, the I^{2}C-bus is assumed to be busy when coming out of a reset, and the BUSY flag in I2Cn_STATUS

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	9	Write data to the transmit buffer	Data is available in transmit buffer
RXDATA	10	Read data from receive buffer	Space is available in receive buffer
None	11	No interaction is required	

Figure 148: I^2C Interactions in Prioritized Order

is thus set. To be able to carry through master operations on the I^{2}C-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^{2}C module detects that the bus is idle can be significant. There are two ways of assuring that the I^{2}C module gets out of the busy state.

- Use the ABORT command in I2Cn_CMD. When the ABORT command is issued, the I^{2}C module is instructed that the bus is idle. The I^{2}C 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.

If operating in slave mode, the above approach is not necessary.

Master Transmitter

To transmit data to a slave, the master must operate as a master transmitter. Table 149 shows the states the I^{2}C 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 I^{2}C 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 to address + the R/W bit is then required by the I^{2}C 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 I^{2}C 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 a data byte by placing it in I2Cn_TXDATA (the master should check the TXBL interrupt flag before writing to I2Cn_TXDATA), this byte 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 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 I^2C, 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.

Master Receiver

To receive data from a slave, the master must operate as a master receiver, see Table 150. 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.

I2Cn_STATE	Description	I2Cn_IF	Required interac-tion	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	
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

Figure 149: I^2C Master Transmitter

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.

I2Cn_STATE	Description	I2Cn_IF	Required interac-tion	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
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

Figure 150: I^2C Master Receiver

P.3.8 Bus States

The I2Cn_STATE register can be used to determine which state the I^2C module and the I^2C bus are in at a given time. The register consists of the STATE bit-field, which shows which state the I^2C 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 I^2C module waiting for a software response.

The possible values of the STATE field are summarized in Table 151. 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 152.

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

Figure 151: I²C STATE Values

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.

Figure 152: I²C Transmission Status

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.

P.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 slave 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, HFPERCLK frequency must be higher than 4.2 MHz for Standard-mode, 11 MHz for Fast-mode, and 24.4 MHz for Fast-mode Plus.

Slave State Machine

The slave state machine is shown in Figure 153. 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.

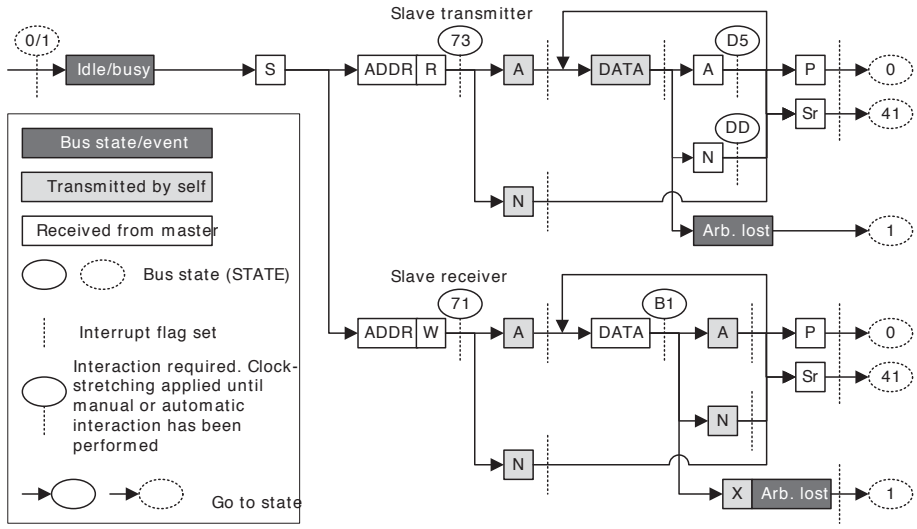


Figure 153: I²C Slave State Machine

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 Section P.3.11. 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.

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 set, the general call address is always accepted regardless of the result of the address recognition. 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.

When an address is accepted by the address comparator, the decision of whether to ACK or NACK the address is passed to software.

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.

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 154 for more information.

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
-	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

Figure 154: I^2C Slave Transmitter

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.

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 155 for more information.

I2Cn_STATE	Description	I2Cn_IF	Required interac-tion	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

Figure 155: I²C - Slave Receiver

P.3.10 Transfer Automation

The I²C can be set up to complete transfers with a minimal amount of interaction.

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.

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.

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 I²C 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^{2}C module always ends a transmission when there is no more data in the transmit buffer. If data has been transmitted on the bus, the transmission is ended after the (N)ACK has been received by the slave. If a START is sent when no data is available in the transmit buffer and AUTOSE is set, then the STOP condition is sent immediately following the START. Software must thus make sure data is available in the transmit buffer before the START condition has been fully transmitted if data is to be transferred.

P.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.

P.3.12 Error Handling

ABORT Command

Some bus errors may require software intervention to be resolved. The I^{2}C module provides an ABORT command, which can be set in I2Cn_CMD, to help resolve bus errors.

When the bus for some reason is locked up and the I^{2}C module is in the middle of a transmission it cannot get out of, or for some other reason the I^{2}C wants to abort a transmission, the ABORT command can be used.

Setting the ABORT command will make the I^{2}C module discard any data currently being transmitted or received, release the SDA and SCL lines and go to an idle mode. ABORT effectively makes the I^{2}C module forget about any ongoing transfers.

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.

I²C-Bus Errors

An I²C-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 I²C-bus. If the I²C 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 156.

Figure 156: I²C Bus Error Response

	Misplaced START	Misplaced STOP
In a master/slave operation	Treated as START. Receive address.	Go idle. Perform any pending actions.

Bus Lockup

A lockup occurs when a master or slave on the I²C-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 I²C-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 Section P.3.12

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.

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 Section P.3.7.

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.

P.3.13 DMA Support

The I^2C module has full DMA support. The DMA controller can write to the transmit buffer using the I2Cn_TXDATA register, and it can read from the receive buffer using the RXDATA register. A request for the DMA controller to read from the I^2C receive buffer can come from the following source:

- Data available in the receive buffer

A write request can come from one of the following sources:

- Transmit buffer and shift register empty. No data to send
- Transmit buffer empty

P.3.14 Interrupts

The interrupts generated by the I^2C module are combined into one interrupt vector, I2C_INT. If I^2C 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.

P.3.15 Wake-up

The I^{2}C receive section can be active all the way down to energy mode EM3, and can wake up the CPU on address interrupt. All address match modes are supported.

P.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	Receive Buffer Data Register
0x020	I2Cn_RXDATAP	R	Receive Buffer Data Peek Register
0x024	I2Cn_TXDATA	W	Transmit Buffer Data Register
0x028	I2Cn_IF	R	Interrupt Flag Register
0x02C	I2Cn_IFS	W1	Interrupt Flag Set Register
0x030	I2Cn_IFC	W1	Interrupt Flag Clear Register
0x034	I2Cn_IEN	RW	Interrupt Enable Register
0x038	I2Cn_ROUTE	RW	I/O Routing Register

P.5 Register Description

P.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
Access														RW			RW		RW				RW			RW	RW	RW	RW	RW	RW	RW
Name														CLTO			GIBITO		BITO				CLHR			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.

Bit	Name	Reset	Access	Description
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.

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	320PPC	Timeout after 320 prescaled clock cycles. In standard mode at 100 kHz, this results in a 400us timeout.
5	1024PPC	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.		
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.

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.

Bit	Name	Reset	Access	Description
11:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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

Bit	Name	Reset	Access	Description
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.
1	When a general call address is received, a software response is required.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
3	AUTOSE	0	RW	**Automatic STOP when Empty**

Bit	Name	Reset	Access	Description
	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.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
0	EN	0	RW	**I^2C Enable**
	Use this bit to enable or disable the I^2C module.			

Value	Description
0	The I^2C module is disabled. And its internal state is cleared
1	The I^2C module is enabled.

P.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.		
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.			

P.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.		
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.

P.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	1	0	0	0	0	0	0	0
Access																								R	R	R	R	R	R	R	R	R
Name																								RXDATAV	TXBL	TXC	PABORT	PCONT	PNACK	PACK	PSTOP	PSTART

Bit	Name	Reset	Access	Description
31:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8	RXDATAV	0	R	**RX Data Valid**

Set when data is available in the receive buffer. Cleared when the receive buffer is empty.

Bit	Name	Reset	Access	Description
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.			

P.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.		
8:0	DIV	0x000	RW	**Clock Divider**
	Specifies the clock divider for the I^2C. Note that DIV must be 1 or higher when slave is enabled.			

P.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.		
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.		

P.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.		
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.		

P.5.8 I2Cn_RXDATA - Receive Buffer 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																									0x00							
Access																									R							
Name																									RXDATA							

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:0	RXDATA	0x00	R	**RX Data**
	Use this register to read from the receive buffer. Buffer is emptied on read access.			

P.5.9 I2Cn_RXDATAP - Receive Buffer Data 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																									0x00							
Access																									R							
Name																									RXDATAP							

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:0	RXDATAP	0x00	R	**RX Data Peek**
	Use this register to read from the receive buffer. Buffer is not emptied on read access.			

P.5.10 I2Cn_TXDATA - Transmit Buffer Data Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																									0x00							
Access																									W							
Name																									TXDATA							

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:0	TXDATA	0x00	W	**TX Data**
	Use this register to write a byte to the transmit buffer.			

P.5.11 I2Cn_IF - Interrupt Flag 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	1	0	0	0	0
Access																R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Name																SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK	RXDATAV	TXBL	TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	SSTOP	0	R	**Slave STOP condition Interrupt Flag**
	Set when a STOP condition has been received. Will be set regardless of the EFM32 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.			
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.			
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.			

P.5.12 I2Cn_IFS - Interrupt Flag Set 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
Access																W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1			W1	W1	W1	W1
Name																SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK			TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	SSTOP	0	W1	**Set SSTOP Interrupt Flag**
	Write to 1 to set the SSTOP interrupt flag.			
15	CLTO	0	W1	**Set Clock Low Interrupt Flag**
	Write to 1 to set the CLTO interrupt flag.			
14	BITO	0	W1	**Set Bus Idle Timeout Interrupt Flag**
	Write to 1 to set the BITO interrupt flag.			
13	RXUF	0	W1	**Set Receive Buffer Underflow Interrupt Flag**
	Write to 1 to set the RXUF interrupt flag.			
12	TXOF	0	W1	**Set Transmit Buffer Overflow Interrupt Flag**
	Write to 1 to set the TXOF interrupt flag.			
11	BUSHOLD	0	W1	**Set Bus Held Interrupt Flag**
	Write to 1 to set the BUSHOLD interrupt flag.			
10	BUSERR	0	W1	**Set Bus Error Interrupt Flag**
	Write to 1 to set the BUSERR interrupt flag.			
9	ARBLOST	0	W1	**Set Arbitration Lost Interrupt Flag**
	Write to 1 to set the ARBLOST interrupt flag.			
8	MSTOP	0	W1	**Set MSTOP Interrupt Flag**
	Write to 1 to set the MSTOP interrupt flag.			
7	NACK	0	W1	**Set Not Acknowledge Received Interrupt Flag**
	Write to 1 to set the NACK interrupt flag.			
6	ACK	0	W1	**Set Acknowledge Received Interrupt Flag**
	Write to 1 to set the ACK interrupt flag.			
5:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	TXC	0	W1	**Set Transfer Completed Interrupt Flag**
	Write to 1 to set the TXC interrupt flag.			
2	ADDR	0	W1	**Set Address Interrupt Flag**
	Write to 1 to set the ADDR interrupt flag.			
1	RSTART	0	W1	**Set Repeated START Interrupt Flag**
	Write to 1 to set the RSTART interrupt flag.			
0	START	0	W1	**Set START Interrupt Flag**
	Write to 1 to set the START interrupt flag.			

P.5.13 I2Cn_IFC - Interrupt Flag Clear 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
Access																W1	W1	W1	W1	W1	W1	W1	W1	W1	W1	W1			W1	W1	W1	W1

Offset	Bit Position																													
Name																	SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK		TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	SSTOP	0	W1	**Clear SSTOP Interrupt Flag**
	Write to 1 to clear the SSTOP interrupt flag.			
15	CLTO	0	W1	**Clear Clock Low Interrupt Flag**
	Write to 1 to clear the CLTO interrupt flag.			
14	BITO	0	W1	**Clear Bus Idle Timeout Interrupt Flag**
	Write to 1 to clear the BITO interrupt flag.			
13	RXUF	0	W1	**Clear Receive Buffer Underflow Interrupt Flag**
	Write to 1 to clear the RXUF interrupt flag.			
12	TXOF	0	W1	**Clear Transmit Buffer Overflow Interrupt Flag**
	Write to 1 to clear the TXOF interrupt flag.			
11	BUSHOLD	0	W1	**Clear Bus Held Interrupt Flag**
	Write to 1 to clear the BUSHOLD interrupt flag.			
10	BUSERR	0	W1	**Clear Bus Error Interrupt Flag**
	Write to 1 to clear the BUSERR interrupt flag.			
9	ARBLOST	0	W1	**Clear Arbitration Lost Interrupt Flag**
	Write to 1 to clear the ARBLOST interrupt flag.			
8	MSTOP	0	W1	**Clear MSTOP Interrupt Flag**
	Write to 1 to clear the MSTOP interrupt flag.			
7	NACK	0	W1	**Clear Not Acknowledge Received Interrupt Flag**
	Write to 1 to clear the NACK interrupt flag.			
6	ACK	0	W1	**Clear Acknowledge Received Interrupt Flag**
	Write to 1 to clear the ACK interrupt flag.			
5:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	TXC	0	W1	**Clear Transfer Completed Interrupt Flag**
	Write to 1 to clear the TXC interrupt flag.			
2	ADDR	0	W1	**Clear Address Interrupt Flag**
	Write to 1 to clear the ADDR interrupt flag.			
1	RSTART	0	W1	**Clear Repeated START Interrupt Flag**
	Write to 1 to clear the RSTART interrupt flag.			
0	START	0	W1	**Clear START Interrupt Flag**
	Write to 1 to clear the START interrupt flag.			

P.5.14 I2Cn_IEN - Interrupt 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
Access																RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																SSTOP	CLTO	BITO	RXUF	TXOF	BUSHOLD	BUSERR	ARBLOST	MSTOP	NACK	ACK	RXDATAV	TXBL	TXC	ADDR	RSTART	START

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	SSTOP	0	RW	**SSTOP Interrupt Enable**

Bit	Name	Reset	Access	Description
	Enable interrupt on SSTOP.			
15	CLTO	0	RW	**Clock Low Interrupt Enable**
	Enable interrupt on clock low timeout.			
14	BITO	0	RW	**Bus Idle Timeout Interrupt Enable**
	Enable interrupt on bus idle timeout.			
13	RXUF	0	RW	**Receive Buffer Underflow Interrupt Enable**
	Enable interrupt on receive buffer underflow.			
12	TXOF	0	RW	**Transmit Buffer Overflow Interrupt Enable**
	Enable interrupt on transmit buffer overflow.			
11	BUSHOLD	0	RW	**Bus Held Interrupt Enable**
	Enable interrupt on bus-held.			
10	BUSERR	0	RW	**Bus Error Interrupt Enable**
	Enable interrupt on bus error.			
9	ARBLOST	0	RW	**Arbitration Lost Interrupt Enable**
	Enable interrupt on loss of arbitration.			
8	MSTOP	0	RW	**MSTOP Interrupt Enable**
	Enable interrupt on MSTOP.			
7	NACK	0	RW	**Not Acknowledge Received Interrupt Enable**
	Enable interrupt when not-acknowledge is received.			
6	ACK	0	RW	**Acknowledge Received Interrupt Enable**
	Enable interrupt on acknowledge received.			
5	RXDATAV	0	RW	**Receive Data Valid Interrupt Enable**
	Enable interrupt on receive buffer full.			
4	TXBL	0	RW	**Transmit Buffer level Interrupt Enable**
	Enable interrupt on transmit buffer level.			
3	TXC	0	RW	**Transfer Completed Interrupt Enable**
	Enable interrupt on transfer completed.			
2	ADDR	0	RW	**Address Interrupt Enable**
	Enable interrupt on recognized address.			
1	RSTART	0	RW	**Repeated START condition Interrupt Enable**
	Enable interrupt on transmitted or received repeated START condition.			
0	START	0	RW	**START Condition Interrupt Enable**
	Enable interrupt on transmitted or received START condition.			

P.5.15 I2Cn_ROUTE - I/O Routing 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									0	0
Access																						RW									RW	RW
Name																						LOCATION									SCLPEN	SDAPEN

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the I^2C I/O pins.			

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
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

Q ARM Universal Synchronous Asynchronous Receiver/Transmitter

Q.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.

Q.2 Features

- Asynchronous and synchronous (SPI) communication
- Full duplex and half duplex
- Separate TX/RX enable
- Separate receive / transmit 2-level 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 on USART0
- SmartCard (ISO7816) mode
- I2S mode
- Separate interrupt vectors for receive and transmit interrupts
- Loopback mode
  - Half duplex communication
  - Communication debugging
- PRS RX input

Q.3 Functional Description

An overview of the USART module is shown in Figure 157.

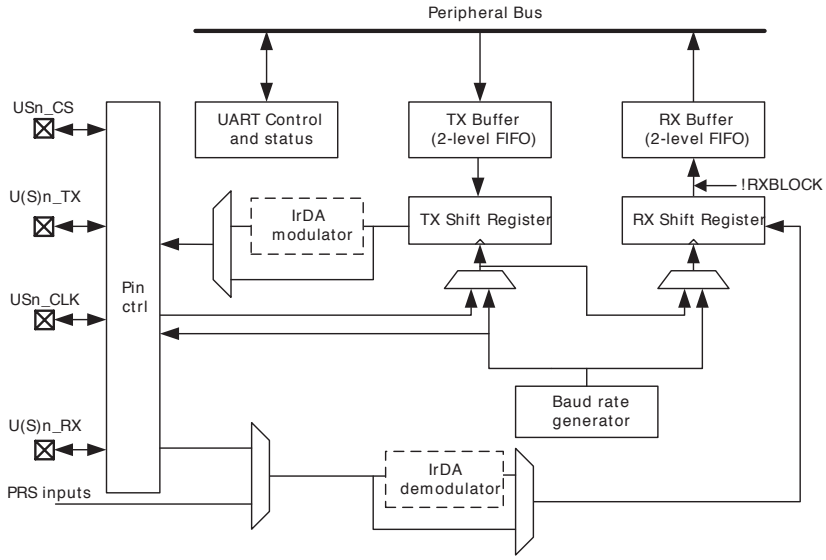


Figure 157: USART Overview

Q.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 158. Full duplex and half duplex communication is supported in both asynchronous and synchronous mode.

Figure 158: 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 159 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 Section Q.3.2 and Section Q.3.3 respectively.

Figure 159: 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]
1	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]

Q.3.2 Asynchronous Operation

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 160.

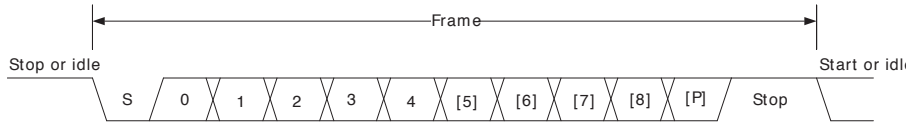


Figure 160: USART Asynchronous Frame Format

The number of data bits in a frame is set by DATABITS in USARTn_FRAME, see Table 161, and the number of stop-bits is set by STOPBITS in USARTn_FRAME, see Table 162. 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.

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

Figure 161: USART Data Bits

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

STOP BITS [1:0]	Number of Stop bits
00	0.5
01	1 (Default)
10	1.5
11	2

Figure 162: USART Stop Bits

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.

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 163. 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.

STOP BITS [1:0]	Description
00	No parity bit (Default)
01	Reserved
10	Even parity
11	Odd parity

Figure 163: USART Parity Bits

Clock Generation

The USART clock defines the transmission and reception data rate. When operating in asynchronous mode, the baud rate (bit-rate) is given by Equation Q.3.2

$$br = f_{HFPERCLK} / (oversample / times (1 + USARTn\_CLKDIV/256)) \quad (8)$$

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 164.

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 15-bit value, with a 13-bit integral part and a 2-bit fractional part. The fractional part is configured in the two LSBs of

OVS [1:0]	oversample
00	16
01	8
10	6
11	4

Figure 164: USART Oversampling

DIV in USART_CLKDIV. The lowest achievable baud rate at 32 MHz is about 244 bauds/sec.

Fractional clock division is implemented by distributing the selected fraction over four baud periods. The fractional part of the divider tells how many of these periods should be extended by one peripheral clock cycle.

Given a desired baud rate $br_{desired}$, the clock divider USARTn_CLKDIV can be calculated by using Equation Q.3.2:

$$\text{USARTn\_CLKDIV} = 256 \times (f_{HFPERCLK} / (\text{oversample} \times br_{desired}) - 1) \quad (9)$$

Table 165 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.

Data Transmission

Asynchronous data transmission is initiated by writing data to the transmit buffer using one of the methods described in Section Q.3.2. 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 transmitter is idle. The TXC status flag is cleared when a new frame becomes available for transmission, but the TXC interrupt flag must be cleared by software.

Desired baud rate [baud/s]	USARTn_OVS =00			USARTn_OVS =01		
	USARTn_CLKDIV/256	Actual baud rate [baud/s]	Error %	USARTn_CLKDIV/256	Actual baud rate [baud/s]	Error %
600	415,75	599,88	-0,02	832,25	600,06	0,01
1200	207,25	1200,48	0,04	415,75	1199,76	-0,02
2400	103,25	2398,082	-0,08	207,25	2400,96	0,04
4800	51	4807,692	0,16	103,25	4796,163	-0,08
9600	25	9615,385	0,16	51	9615,385	0,16
14400	16,25	14492,75	0,64	33,75	14388,49	-0,08
19200	12	19230,77	0,16	25	19230,77	0,16
28800	7,75	28571,43	-0,79	16,25	28985,51	0,64
38400	5,5	38461,54	0,16	12	38461,54	0,16
57600	3,25	58823,53	2,12	7,75	57142,86	-0,79
76800	2,25	76923,08	0,16	5,5	76923,08	0,16
115200	1,25	111111,1	-3,55	3,25	117647,1	2,12
230400	0	250000	8,51	1,25	222222,2	-3,55

Figure 165: USART Baud Rates @ 4MHz Peripheral Clock

Transmit Buffer Operation The transmit-buffer is a 2-level 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 166 shows the basics of the transmit buffer when DATABITS in USARTn_FRAME is configured to less than 10 bits.

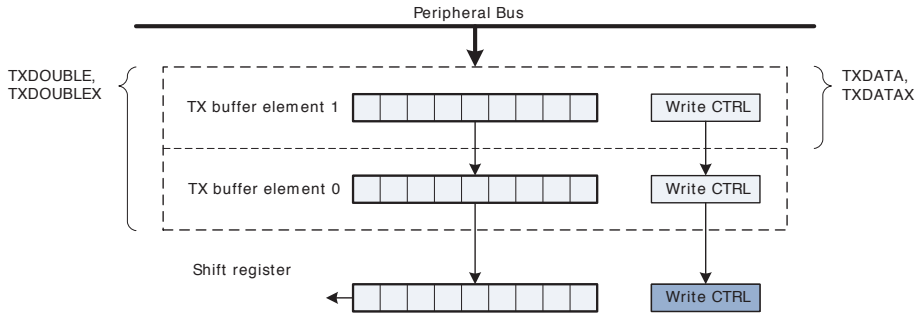


Figure 166: USART Transmit Buffer Operation

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 transmitter is idle, TXBL in USARTn_STATUS and the TXBL interrupt flag in USARTn_IF are used to indicate the level of the transmit buffer. TXBIL in USARTn_CTRL controls the level at which these bits are set. If TXBIL is cleared, they are set whenever the transmit buffer becomes empty, and if TXBIL is set, they are set whenever the transmit buffer goes from full to half-full or empty. Both the TXBL status flag and the TXBL interrupt flag are cleared automatically when their condition becomes false

The transmit buffer, including the transmit shift register can be cleared by setting CLEARTX in USARTn_CMD. This will prevent the USART from transmitting the data in the buffer and shift register, and will make them available for new data. Any frame currently being transmitted will not be aborted. Transmission of this frame will be completed.

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.

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.

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.

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 167.

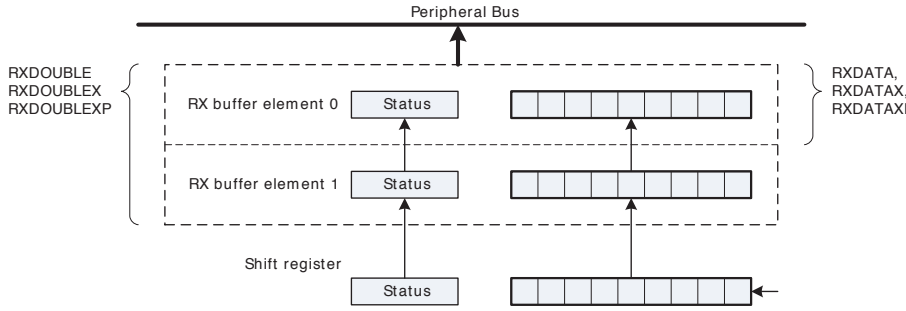


Figure 167: USART Receive Buffer Operation

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.

Blocking Incoming Data When using hardware frame recognition, as detailed in Section Q.3.2 and Section Q.3.2, 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 Section Q.3.2 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.

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.

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 168. 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 168.

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.

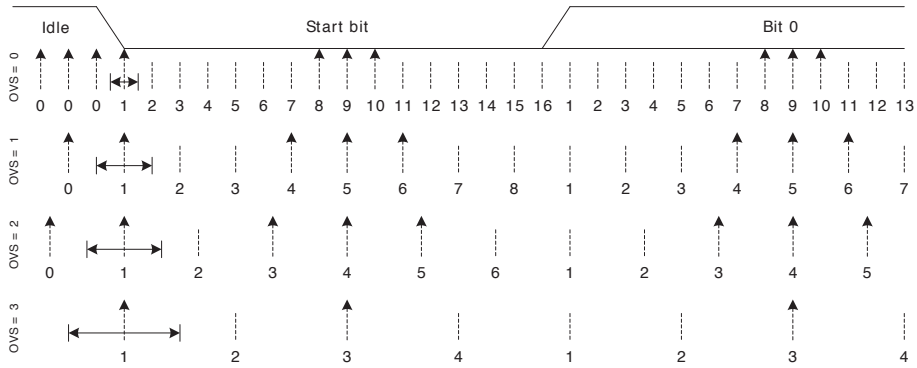


Figure 168: USART Sampling of Start and Data Bits

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 169. 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 169, 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.

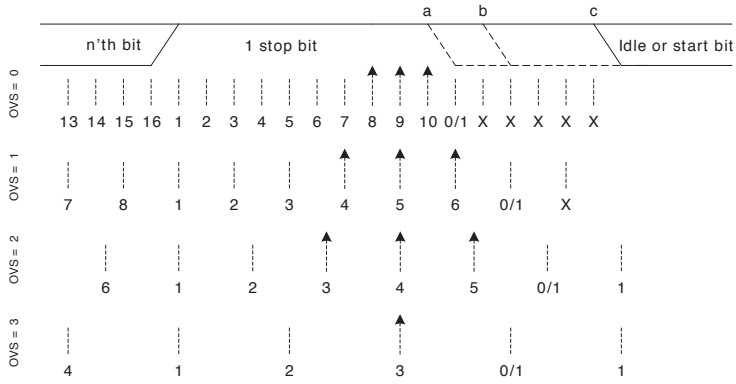


Figure 169: USART Sampling of Stop Bits when Number of Stop Bits are 1 or More

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.

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.

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.

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 170. 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.

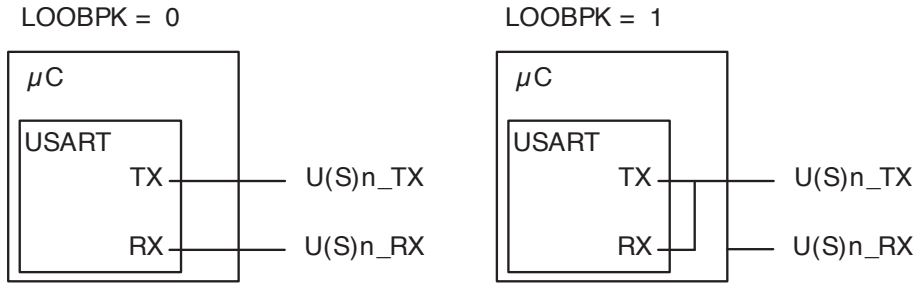


Figure 170: USART Local Loopback

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.

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.

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.

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.

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 one baud period before the transmitter starts transmitting data, and deactivated when the last bit has been transmitted and there is no more data in the transmit buffer to transmit, or the transmitter becomes disabled. This feature can be used to turn the external driver on when transmitting data, and turn it off when the data has been transmitted.

Figure 171 shows an example configuration where USn_CS is used to automatically enable and disable an external driver.

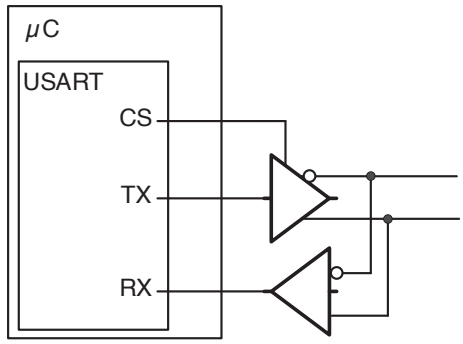


Figure 171: USART Half Duplex Communication with External Driver

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).

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.

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 172. The first element in the transmit buffer, i.e. element 0 in Figure 172 is the first element written to the FIFO, or the least significant byte when writing two bytes at a time using USARTn_TXDOUBLE.

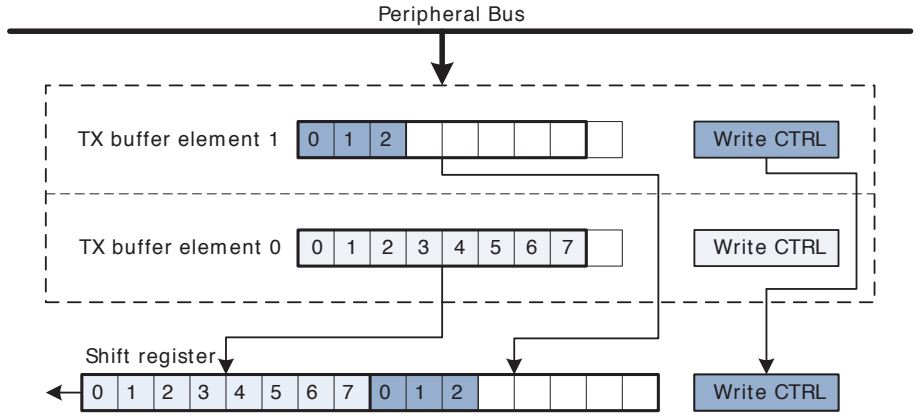


Figure 172: USART Transmission of Large Frames

As shown in Figure 172, 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.

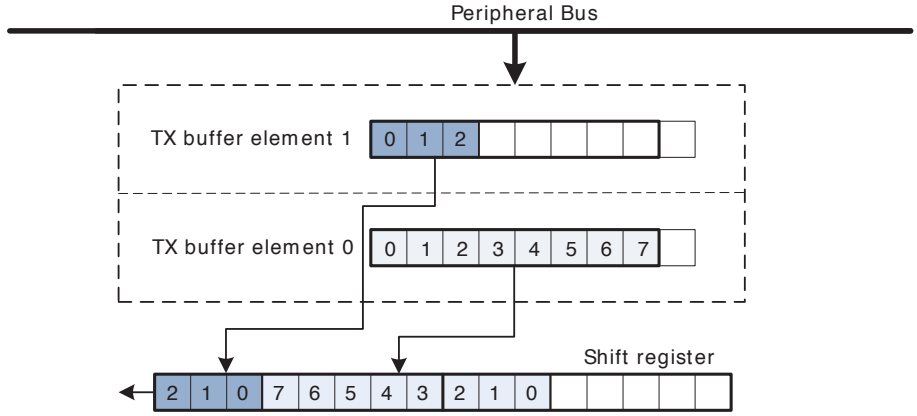


Figure 173: USART Transmission of Large Frames, MSBF

Figure 173 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 174. 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.

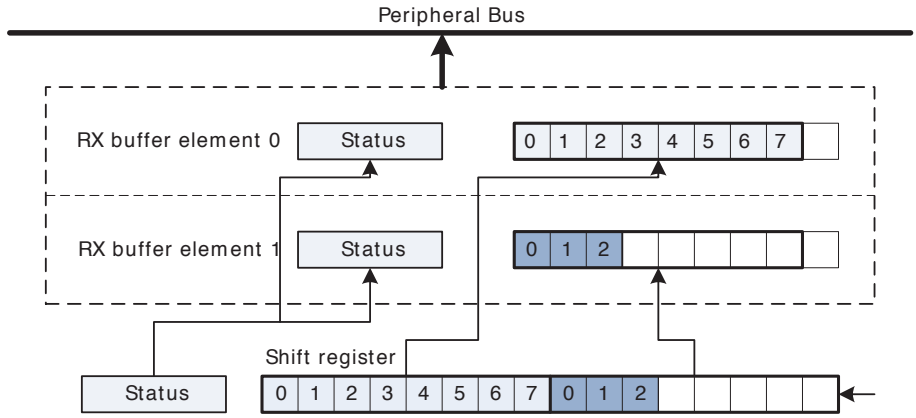


Figure 174: USART Reception of Large Frames

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.

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.

Example Q.3.2 explains basic usage of the multi-processor mode:

exampleUSART Multi-processor Mode Example

1. All slaves enable multi-processor mode and, enable and block the receiver. They will now not receive data unless it is an address frame. MPAB in 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.

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.

SmartCard Mode

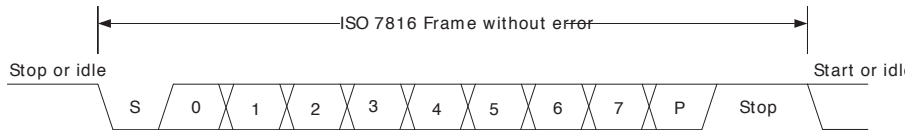
In SmartCard mode, the USART supports the ISO 7816 I/O line T0 mode. With exception of the stop-bits (guard time), the 7816 data frame is equal to the regular asynchronous frame. In this mode, the receiver pulls the line low for one baud, half a baud into the guard time to indicate a parity error. This 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.

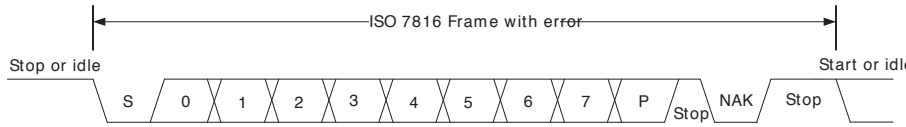
When no parity error is identified by the receiver, the data frame is as shown in Figure 175. The frame consists of 8 data bits, a parity bit, and 2 stop bits. The transmitter does not drive the output line during the guard time.

Figure 175: 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 176. 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 176: 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 177. 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.

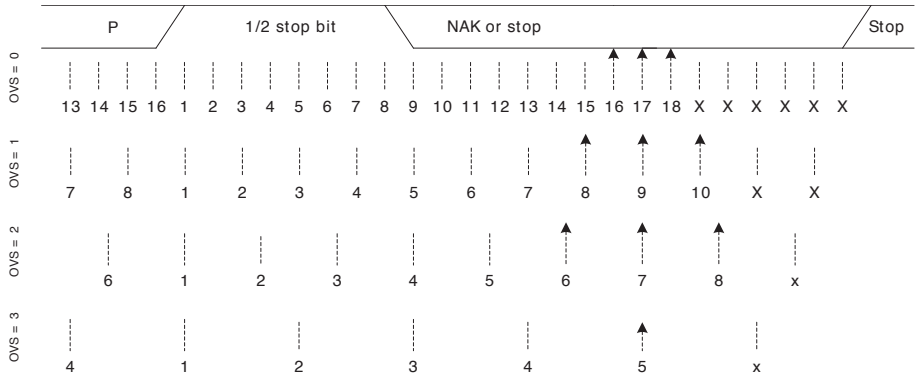


Figure 177: USART SmartCard Stop Bit Sampling

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.

Q.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.

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.

Clock Generation

The bit-rate in synchronous mode is given by Equation Q.3.3. As in the case of asynchronous operation, the clock division factor have a 13-bit integral part and a 2-bit fractional part.

$$br = f_{HFPERCLK}/(2 \times (1 + USARTn\_CLKDIV/256)) \quad (10)$$

Given a desired baud rate brdesired, the clock divider USARTn_CLKDIV can be calculated using Equation Q.3.3

$$USARTn\_CLKDIV = 256 \times (f_{HFPERCLK}/(2 \times brdesired) - 1) \quad (11)$$

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 178. Figure 179 shows the resulting timing of data set-up and sampling relative to the bus clock.

Figure 178: 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

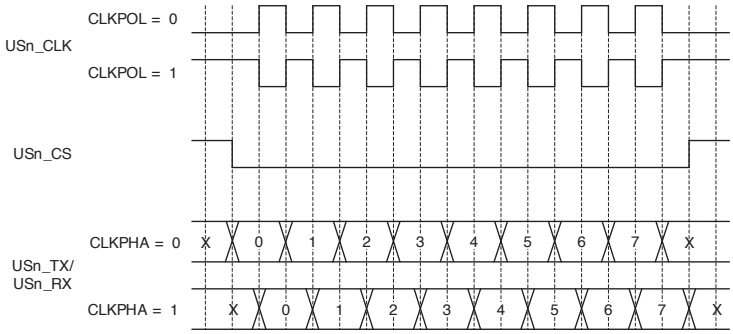


Figure 179: USART SPI Timing

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.

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.

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 when a transmission begins, and deactivated directly after the last bit has been transmitted and there is no more data in the transmit buffer. 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.

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.

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.

Synchronous Half Duplex Communication

Half duplex communication in synchronous mode is very similar to half duplex communication in asynchronous mode as detailed in Section Q.3.2. 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.

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.

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.

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.

Major Modes The USART supports a set of different I2S formats as shown in Table 180, 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 183 and Figure 184. 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.

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

Figure 180: USART I2S Modes

The regular I2S waveform is shown in Figure 181 and Figure 182. 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

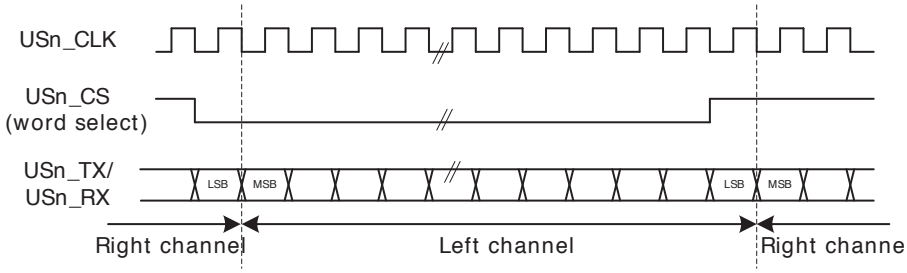


Figure 181: USART Standard I2S waveform

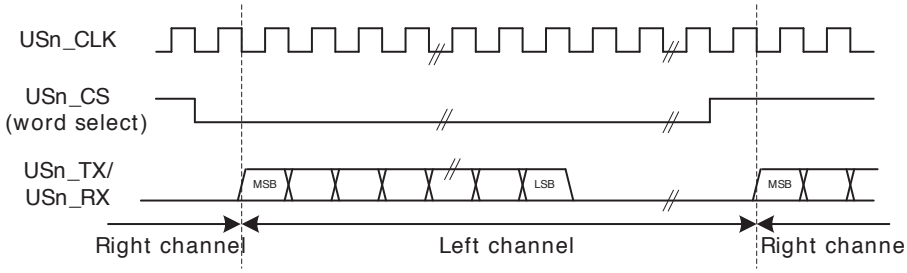


Figure 182: USART Standard I2S waveform (reduced accuracy)

A left-justified stream is shown in Figure 183. 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.

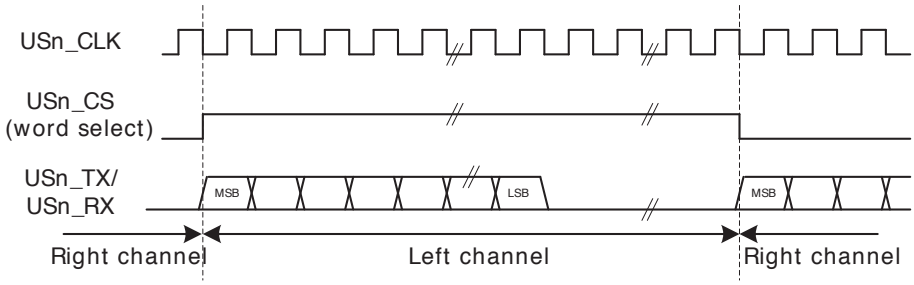


Figure 183: USART Left-justified I2S waveform

A right-justified stream is shown in Figure 184. The left and right justified streams are equal when the data-size is equal to the word-width.

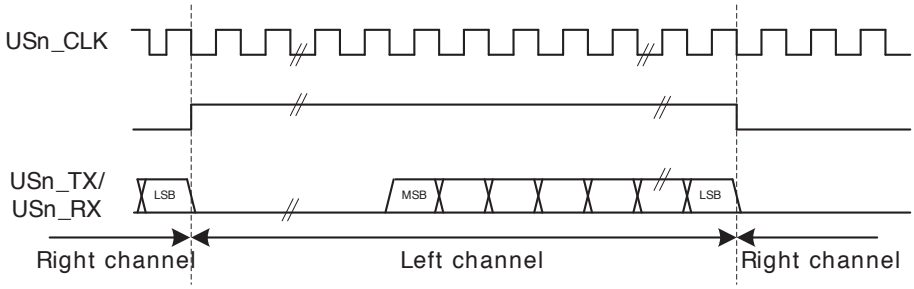


Figure 184: USART Right-justified I2S waveform

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 185.

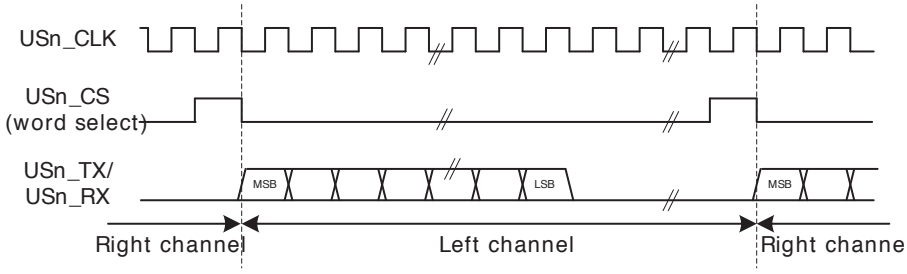


Figure 185: USART Mono I2S waveform

Using I2S Mode When using the USART in I2S mode, DATABITS in USARTn_FRAME must be set to 8 or 16 data-bits. 8 databits can be used in all modes, and 16 can be used in the modes where the number of bytes in the I2S word is even. In addition to this, MSBF in USARTn_CTRL should be set, and CLKPOL and CLKPHA in USARTn_CTRL should be cleared.

The USART does not have separate TX and RX buffers for left and right data, so when using I2S in stereo mode, the application must keep track of whether the

buffers contain left or right data. This can be done by observing TXBLRIGHT, RXDATAVRIGHT and RXFULLRIGHT in USARTn_STATUS. TXBLRIGHT tells whether TX is expecting data for the left or right channel. It will be set with TXBL if right data is expected. The receiver will set RXDATAVRIGHT if there is at least one right element in the buffer, and RXFULLRIGHT if the buffer is full of right elements.

When using I2S with DMA, separate DMA requests can be used for left and right data by setting DMASPLIT in USARTn_I2SCTRL.

In both master and slave mode the USART always starts transmitting on the LEFT channel after being enabled. In master mode, the transmission will stop if TX becomes empty. In that case, TXC is set. Continuing the transmission in this case will make the data-stream continue where it left off. To make the USART start on the LEFT channel after going empty, disable and re-enable TX.

Q.3.4 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

Q.3.5 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.

Q.3.6 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.
- 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.

Q.3.7 Transmission Delay

By configuring TXDELAY in USARTn_CTRL, the transmitter can be forced to wait a number of bit-periods from it is ready to transmit data, to 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 baud 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.

TXDELAY in USARTn_CTRL only applies to asynchronous transmission.

Q.3.8 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

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

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.

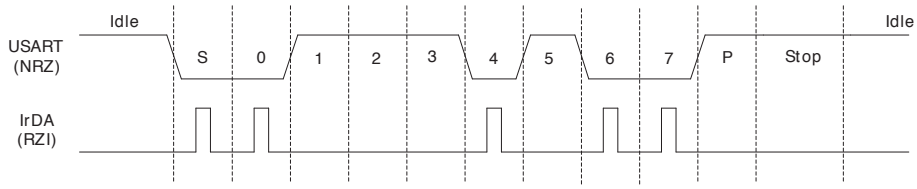
Q.3.9 IrDA Modulator/Demodulator

The IrDA modulator on USART0 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 USART0. In the same way, the input signal is demodulated before it enters the actual USART module. The modulator is only available on USART0, and 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 186.

Figure 186: 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 187.

Figure 187: 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.

Q.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	RW	Clock Control Register
0x018	USARTn_RXDATAX	R	RX Buffer Data Extended Register
0x01C	USARTn_RXDATA	R	RX Buffer Data Register
0x020	USARTn_RXDOUBLEX	R	RX Buffer Double Data Extended Register
0x024	USARTn_RXDOUBLE	R	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	W1	Interrupt Flag Clear Register
0x04C	USARTn_IEN	RW	Interrupt Enable Register
0x050	USARTn_IRCTRL	RW	IrDA Control Register
0x054	USARTn_ROUTE	RW	I/O Routing Register
0x058	USARTn_INPUT	RW	USART Input Register
0x05C	USARTn_I2SCTRL	RW	I2S Control Register

Q.5 Register Description

Q.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	0x0		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	RW
Name	SMSDELAY	MVDIS	AUTOTX	BYTESWAP	TXDELAY		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

Bit	Name	Reset	Access	Description
27:26	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 baud period
2	DOUBLE	Transmission of new frames are delayed by two baud periods
3	TRIPLE	Transmission of new frames are delayed by three baud periods

Bit	Name	Reset	Access	Description
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

Bit	Name	Reset	Access	Description
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
1	Framing and parity errors disable the receiver

Bit	Name	Reset	Access	Description
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

Bit	Name	Reset	Access	Description
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.			

Bit	Name	Reset	Access	Description

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
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
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.

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.

Bit	Name	Reset	Access	Description

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

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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

Bit	Name	Reset	Access	Description
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

Bit	Name	Reset	Access	Description
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

Bit	Name	Reset	Access	Description
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
1	The USART operates in synchronous mode

Q.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.		
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.		
9:8	PARITY	0x0	RW	**Parity-Bit Mode**

Bit	Name	Reset	Access	Description

Determines whether parity bits are enabled, and whether even or odd parity should be used. Only available in asynchronous mode.

Value	Mode	Description
0	NONE	Parity bits are not used
2	EVEN	Even parity are used. Parity bits are automatically generated and checked by hardware.
3	ODD	Odd parity is used. Parity bits are automatically generated and checked by hardware.

7:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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
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

Q.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																										0	0	0		0x0		
Access																										RW	RW	RW		RW		
Name																										AUTOTXTEN	TXTEN	RXTEN		TSEL		

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	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.

Bit	Name	Reset	Access	Description
	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

Q.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.		
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.			

Q.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																				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
Name																				RXFULLRIGHT	RXDATAVRIGHT	TXBSRIGHT	TXBDRIGHT	RXFULL	RXDATAV	TXBL	TXC	TXTRI	RXBLOCK	MASTER	TXENS	RXENS

Bit	Name	Reset	Access	Description
31:13	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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 cleared, TXBL is set whenever the transmit buffer is empty, and if TXBIL is set, TXBL is set whenever the transmit buffer is half-full or empty.			
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.			

Q.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												0x0000																				
Access												RW																				
Name												DIV																				

Bit	Name	Reset	Access	Description
31:21	Reserved	To ensure compatibility with future devices, always write bits to 0.		

Bit	Name	Reset	Access	Description
20:6	DIV	0x0000	RW	**Fractional Clock Divider**
	Specifies the fractional clock divider for the USART.			
5:0	Reserved	To ensure compatibility with future devices, always write bits to 0.		

Q.5.7 USARTn_RXDATAX - RX Buffer Data Extended 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						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.		
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.		
8:0	RXDATA	0x000	R	**RX Data**
	Use this register to access data read from the USART. Buffer is cleared on read access.			

Q.5.8 USARTn_RXDATA - RX Buffer 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																									0x00							
Access																									R							
Name																									RXDATA							

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

Q.5.9 USARTn_RXDOUBLEX - RX Buffer Double Data Extended Register

Offset	Bit Position																														
0x020	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	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.		
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.		
8:0	RXDATA0	0x000	R	**RX Data 0**
	First frame read from buffer.			

Q.5.10 USARTn_RXDOUBLE - RX FIFO Double 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																	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.		
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.			

Q.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.		
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.		
8:0	RXDATAP	0x000	R	**RX Data Peek**

Bit	Name	Reset	Access	Description
	Use this register to access data read from the USART.			

Q.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					0x000								
Access	R	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.		
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.		
8:0	RXDATAP0	0x000	R	**RX Data 0 Peek**
	First frame read from FIFO.			

Q.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.		
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 WDATA.			
12	TXTRIAT	0	W	**Set TXTRI After Transmission**
	Set to tristate transmitter by setting TXTRI after transmission.			

Bit	Name	Reset	Access	Description
11	UBRXAT	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.		
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.			

Q.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.		
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.			

Q.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_WDATA.			
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.		
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.			

Bit	Name	Reset	Access	Description
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 WDATA.			
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.		
8:0	TXDATA0	0x000	W	**TX Data**
	First frame to write to buffer.			

Q.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.		
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.			

Q.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	1	0
Access	R																			R	R	R	R	R	R	R	R	R	R	R	R	R
Name																				CCF	SSM	MPAF	FERR	PERR	TXUF	TXOF	RXUF	RXOF	RXFULL	RXDATAV	TXBL	TXC

Bit	Name	Reset	Access	Description
31:13	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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**

Bit	Name	Reset	Access	Description
	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.			
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 TXBIL is set, or when buffer goes from full to half-full if TXBIL is cleared.			
0	TXC	0	R	**TX Complete Interrupt Flag**
	This interrupt is used after a transmission when both the TX buffer and shift register are empty.			

Q.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
Access																				W1	W1	W1	W1	W1	W1	W1	W1	W1	W1			W1
Name																				CCF	SSM	MPAF	FERR	PERR	TXUF	TXOF	RXUF	RXOF	RXFULL			TXC

Bit	Name	Reset	Access	Description
31:13	Reserved	To ensure compatibility with future devices, always write bits to 0.		
12	CCF	0	W1	**Set Collision Check Fail Interrupt Flag**
	Write to 1 to set the CCF interrupt flag.			
11	SSM	0	W1	**Set Slave-Select in Master mode Interrupt Flag**
	Write to 1 to set the SSM interrupt flag.			
10	MPAF	0	W1	**Set Multi-Processor Address Frame Interrupt Flag**
	Write to 1 to set the MPAF interrupt flag.			
9	FERR	0	W1	**Set Framing Error Interrupt Flag**
	Write to 1 to set the FERR interrupt flag.			
8	PERR	0	W1	**Set Parity Error Interrupt Flag**
	Write to 1 to set the PERR interrupt flag.			
7	TXUF	0	W1	**Set TX Underflow Interrupt Flag**
	Write to 1 to set the TXUF interrupt flag.			
6	TXOF	0	W1	**Set TX Overflow Interrupt Flag**
	Write to 1 to set the TXOF interrupt flag.			
5	RXUF	0	W1	**Set RX Underflow Interrupt Flag**
	Write to 1 to set the RXUF interrupt flag.			
4	RXOF	0	W1	**Set RX Overflow Interrupt Flag**
	Write to 1 to set the RXOF interrupt flag.			
3	RXFULL	0	W1	**Set RX Buffer Full Interrupt Flag**
	Write to 1 to set the RXFULL interrupt flag.			
2:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	TXC	0	W1	**Set TX Complete Interrupt Flag**
	Write to 1 to set the TXC interrupt flag.			

Q.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
Access																				W1	W1	W1	W1	W1	W1	W1	W1	W1	W1			W1
Name																				CCF	SSM	MPAF	FERR	PERR	TXUF	TXOF	RXUF	RXOF	RXFULL			TXC

Bit	Name	Reset	Access	Description
31:13	Reserved	To ensure compatibility with future devices, always write bits to 0.		
12	CCF	0	W1	**Clear Collision Check Fail Interrupt Flag**
	Write to 1 to clear the CCF interrupt flag.			
11	SSM	0	W1	**Clear Slave-Select In Master Mode Interrupt Flag**
	Write to 1 to clear the SSM interrupt flag.			
10	MPAF	0	W1	**Clear Multi-Processor Address Frame Interrupt Flag**
	Write to 1 to clear the MPAF interrupt flag.			
9	FERR	0	W1	**Clear Framing Error Interrupt Flag**
	Write to 1 to clear the FERR interrupt flag.			
8	PERR	0	W1	**Clear Parity Error Interrupt Flag**
	Write to 1 to clear the PERR interrupt flag.			
7	TXUF	0	W1	**Clear TX Underflow Interrupt Flag**
	Write to 1 to clear the TXUF interrupt flag.			
6	TXOF	0	W1	**Clear TX Overflow Interrupt Flag**
	Write to 1 to clear the TXOF interrupt flag.			
5	RXUF	0	W1	**Clear RX Underflow Interrupt Flag**
	Write to 1 to clear the RXUF interrupt flag.			
4	RXOF	0	W1	**Clear RX Overflow Interrupt Flag**
	Write to 1 to clear the RXOF interrupt flag.			
3	RXFULL	0	W1	**Clear RX Buffer Full Interrupt Flag**
	Write to 1 to clear the RXFULL interrupt flag.			
2:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	TXC	0	W1	**Clear TX Complete Interrupt Flag**
	Write to 1 to clear the TXC interrupt flag.			

Q.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
Access																				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Name																				CCF	SSM	MPAF	FERR	PERR	TXUF	TXOF	RXUF	RXOF	RXFULL	RXDATAV	TXBL	TXC

Bit	Name	Reset	Access	Description
31:13	Reserved	To ensure compatibility with future devices, always write bits to 0.		
12	CCF	0	RW	**Collision Check Fail Interrupt Enable**
	Enable interrupt on collision check error detected.			
11	SSM	0	RW	**Slave-Select In Master Mode Interrupt Enable**

Bit	Name	Reset	Access	Description
	Enable interrupt on slave-select in master mode.			
10	MPAF	0	RW	**Multi-Processor Address Frame Interrupt Enable**
	Enable interrupt on multi-processor address frame.			
9	FERR	0	RW	**Framing Error Interrupt Enable**
	Enable interrupt on framing error.			
8	PERR	0	RW	**Parity Error Interrupt Enable**
	Enable interrupt on parity error (asynchronous mode only).			
7	TXUF	0	RW	**TX Underflow Interrupt Enable**
	Enable interrupt on TX underflow.			
6	TXOF	0	RW	**TX Overflow Interrupt Enable**
	Enable interrupt on TX overflow.			
5	RXUF	0	RW	**RX Underflow Interrupt Enable**
	Enable interrupt on RX underflow.			
4	RXOF	0	RW	**RX Overflow Interrupt Enable**
	Enable interrupt on RX overflow.			
3	RXFULL	0	RW	**RX Buffer Full Interrupt Enable**
	Enable interrupt on RX Buffer full.			
2	RXDATAV	0	RW	**RX Data Valid Interrupt Enable**
	Enable interrupt on RX data.			
1	TXBL	0	RW	**TX Buffer Level Interrupt Enable**
	Enable interrupt on TX buffer level.			
0	TXC	0	RW	**TX Complete Interrupt Enable**
	Enable interrupt on TX complete.			

Q.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																									0	0x0			0	0x0		0
Access																									RW	RW			RW	RW		RW
Name																									IRPRSEN	IRPRSSEL			IRFILT	IRPW		IREN

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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	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

3	IRFILT	0	RW	**IrDA RX Filter**
	Set to enable filter on IrDA demodulator.			

Bit	Name	Reset	Access	Description
	Value	**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.			

Q.5.22 USARTn_ROUTE - I/O Routing Register

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

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the USART I/O 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	
7:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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		
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		
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		
0	RXPEN	0	RW	**RX Pin Enable**

Bit	Name	Reset	Access	Description
	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

Q.5.23 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			
Access																												RW	RW			
Name																												RXPRS	RXPRSSEL			

Bit	Name	Reset	Access	Description
31:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4	RXPRS	0	RW	**PRS RX Enable**
	When set, the PRS channel selected as input to RX.			
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

Q.5.24 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.		
10:8	FORMAT	0x0	RW	**I2S Word Format**

Bit	Name	Reset	Access	Description
	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

Bit	Name	Reset	Access	Description
7:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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			
0	EN	0	RW	**Enable I2S Mode**
	Set the U(S)ART in I2S mode.			

R ARM Universal Asynchronous Receiver/Transmitter

R.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.

R.2 Features

- Full duplex and half duplex
- Separate TX / RX enable
- Separate receive / transmit 2-level buffers, with additional separate shift registers
- Programmable baud rate, generated as an fractional division from the peripheral clock (HFPERCLK)
- Max bit-rate
  - UART standard mode, peripheral clock rate / 16
  - UART FAST mode, peripheral clock rate / 8
- 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 can trigger transmissions
- Full DMA support
- PRS RX input

R.3 Functional Description

The UART is functionally equivalent to the USART with the exceptions defined in Table 188. 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.

Figure 188: UART Limitations

Feature	Limitations
Synchronous operation	Not available. SYNC, CSMA, 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.
Frame size	Limited to 8-9 databits. Other configurations of DATABITS in USARTn_FRAME are not possible.
IrDA	Not available. IREN in USARTn_IRCTRL is always 0.

R.4 Register Description

The register description of the UART is equivalent to the register description of the USART except the limitations mentioned in Table 188. See the USART chapter for complete information.

R.5 Register Map

The register map of the UART is equivalent to the register map of the USART. See the USART chapter for complete information.

S ARM Low Energy Universal Asynchronous Receiver/Transmitter

S.1 Introduction

The unique LEUART™, the Low Energy UART, 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 at baud rates up to 9600.

Even when the EFM is in low energy mode EM2 (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 to indicate e.g. the end of a data transmission. The start frame and signal frame can be used in combination for instance to handle higher level communication protocols.

Similarly, data can be transmitted in EM2 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 minimum of software intervention and energy consumption.

S.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/s to 9600 baud/s
- 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, EM1 and EM2 with
  - Full DMA support
  - Specified start-byte can start reception automatically
- IrDA modulator (pulse generator, pulse extender)
- Multi-processor mode
- Loopback mode
  - Half duplex communication
  - Communication debugging
- PRS RX input

S.3 Functional Description

An overview of the LEUART module is shown in Figure 189.

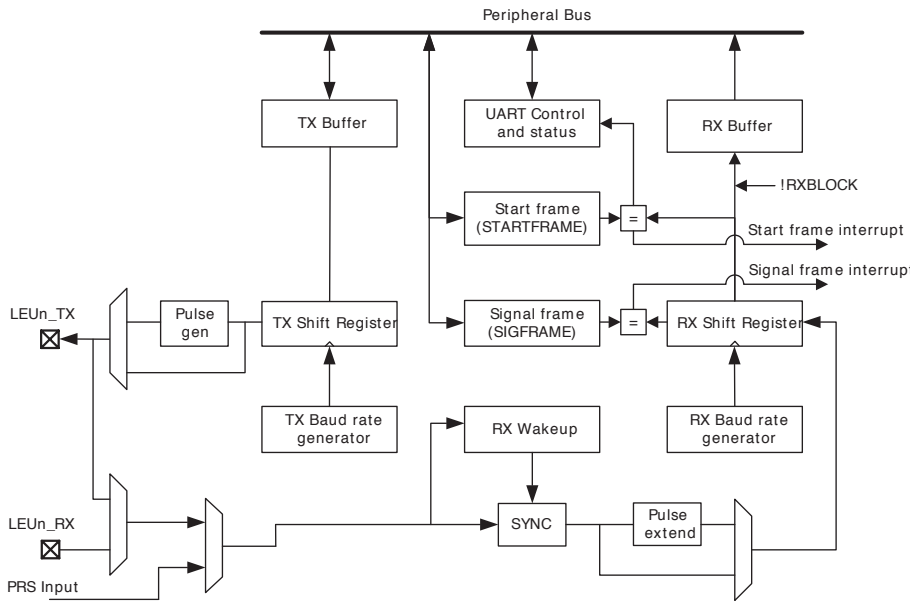


Figure 189: LEUART Overview

S.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 190.

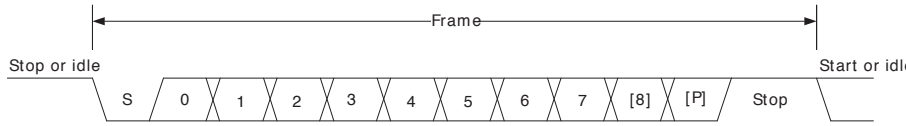


Figure 190: LEUART Asynchronous Frame Format

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.

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 191. 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.

PARITY [1:0]	Description
00	No parity (default)
01	Reserved
10	Even parity
11	Odd parity

Figure 191: LEUART Parity Bit

See Section S.3.5 for more information on parity bit handling.

S.3.2 Clock Source

The LEUART clock source is selected by the LFB bit field the CMU_LFCLKSEL register. The clock is prescaled by the LEUARTn bitfield in the CMU_LFBPRESC0 register and enabled by the LEUARTn bit in the CMU_LFBCLKEN0.

To use this module, the LE interface clock must be enabled in CMU_HFCORECLKEN0, in addition to the module clock.

S.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 12-bit value, with a 7-bit integral part and a 5-bit fractional part. The baud rate of the LEUART is given by :

br = fLEUARTn/(1 + LEUARTn_CLKDIV/256)(12)

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.

For a desired baud rate $br_{DESIRED}$, LEUARTn_CLKDIV can be calculated by using:

LEUARTn_CLKDIV = 256 $\times$(fLEUARTn/$br_{DESIRED-1)}$ (13)

Table 192 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.

Figure 192: LEUART Baud Rates

Desired baud rate [baud/s]	LEUARTn_CLKDIV	LEUARTn_CLKDIV/256	Actual baud rate [baud/s]	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

S.3.4 Data Transmission

Data transmission is initiated by writing data to the transmit buffer using one of the methods described in Section S.3.4. 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 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.

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.

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 193.

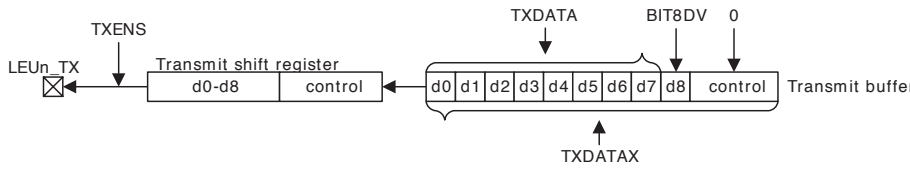


Figure 193: LEUART Transmitter Overview

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 WBREAK, 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 baud 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 however, if AUTOTRI in LEUARTn_CTRL is set. See Section S.3.7 for more information on half duplex operation.

Jitter in Transmitted Data

Internally the LEUART module uses only the positive edges of the 32.768 kHz clock (LFBCLK) for transmission and reception. Transmitted data will thus have jitter equal to the difference between the optimal data set-up location and the closest positive edge on the 32.768 kHz clock. The jitter in on the location data is set up by the transmitter will thus be no more than half a clock period according to the optimal set-up location. The jitter in the period of a single baud output by the transmitter will never be more than one clock period.

S.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 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 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.

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 latter 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 194.

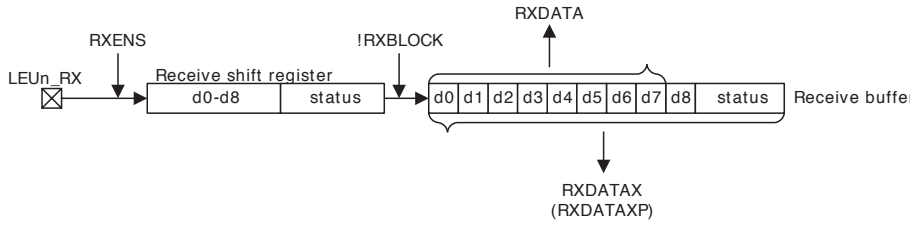


Figure 194: LEUART Receiver Overview

Blocking Incoming Data

When using hardware frame recognition, as detailed in Section S.3.5, Section S.3.5, and Section S.3.5, 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 Section S.3.5. The other case is when receiving a start-frame when SFUBRX in LEUARTn_CTRL is set; see Section S.3.5

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.

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.

Data Sampling

The receiver samples each incoming baud as close as possible to the middle of the baud-period. Except for the start-bit, only a single sample is taken of each of the incoming bauds.

The length of a baud-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 baud in the UART frame is then given by the following equation:

$$S_{opt}(n) = n\ (1 + \text{LEUARTn\_CLKDIV}/256) + \text{CLKDIV}/512 \quad (14)$$

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:

$$S(n) = \text{floor}(n \times (1 + \text{LEUARTn\_CLKDIV}/256) + \text{LEUARTn\_CLKDIV}/512) \quad (15)$$

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.

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.

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.

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.

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.

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, 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.

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.

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.

S.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 195. 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.

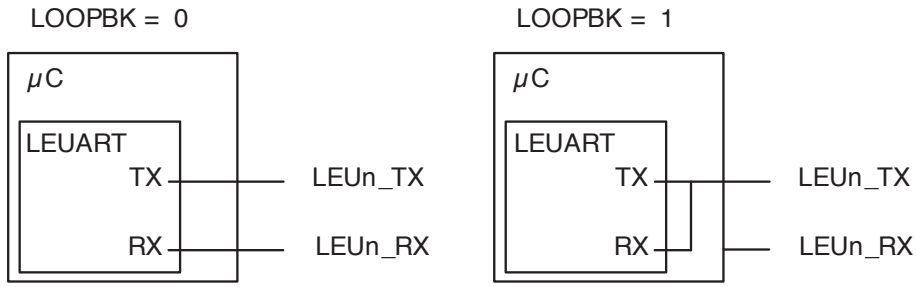


Figure 195: LEUART Local Loopback

S.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.

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.

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.

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 196 shows an example configuration using an external driver.

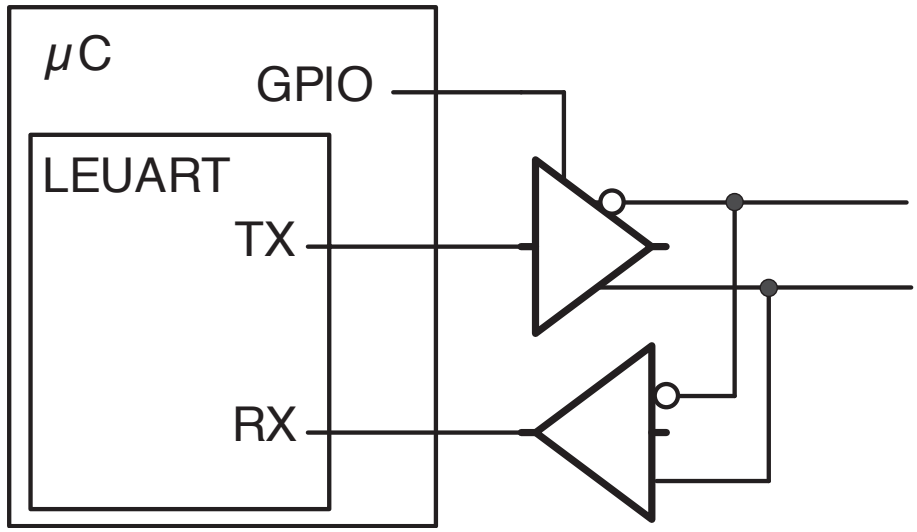


Figure 196: LEUART Half Duplex Communication with External Driver

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.

S.3.8 Transmission Delay

By configuring TXDELAY in LEUARTn_CTRL, the transmitter can be forced to wait a number of bit-periods from it is ready to transmit data, to 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 baud 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.

S.3.9 PRS RX Input

The LEUART can be configured to receive data directly from the PRS channel by setting RX_PRS in LEUARTn_INPUT. The PRS channel used can be selected using RX_PRS_SEL in LEUARTn_INPUT.

S.3.10 DMA Support

The LEUART has full DMA support in energy modes EM0 - EM2. 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, 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.

When RXDMAWU or TXDMAWU is set, the system will not be able to go to EM2/EM3 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/EM3 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.

S.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 197.

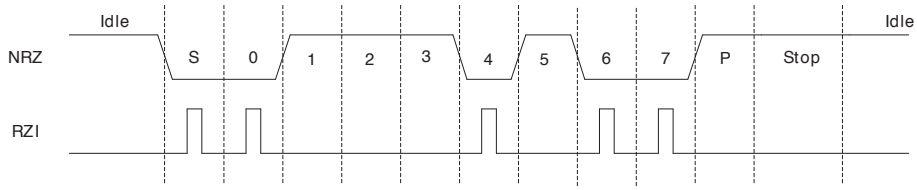


Figure 197: LEUART - NRZ vs. RZI

If PULSEEN in LEUARTn_PULSECTRL is set while INV in LEUARTn_CTRL is cleared, the output waveform will like RZI shown in Figure 197, 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 baud period.

At 2400 baud/s 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.

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.

S.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. Please refer to [?] for a description on how to perform register accesses to Low Energy Peripherals.

S.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	Receive Buffer Data Extended Register
0x01C	LEUARTn_RXDATA	R	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	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_ROUTE	RW	I/O Routing Register
0x0AC	LEUARTn_INPUT	RW	LEUART Input Register

S.5 Register Description

S.5.1 LEUARTn_CTRL - Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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

Offset	Bit Position															
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.		
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 baud period
2	DOUBLE	Transmission of new frames are delayed by two baud periods
3	TRIPLE	Transmission of new frames are delayed by three baud 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.

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.

Bit	Name	Reset	Access	Description
	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 0, 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**
	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		

S.5.2 LEUARTn_CMD - Command Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
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**

Bit	Name	Reset	Access	Description
	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.			

S.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																											0	1	0	0	0	0
Access																											R	R	R	R	R	R
Name																											RXDATAV	TXBL	TXC	RXBLOCK	TXENS	RXENS

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

S.5.4 LEUARTn_CLKDIV - Clock Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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																		0x000														
Access																		RW														
Name																		DIV														

Bit	Name	Reset	Access	Description
31:15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14:3	DIV	0x000	RW	**Fractional Clock Divider**
	Specifies the fractional clock divider for the LEUART.			
2:0	Reserved	To ensure compatibility with future devices, always write bits to 0.		

S.5.5 LEUARTn_STARTFRAME - Start Frame Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
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.			

S.5.6 LEUARTn_SIGFRAME - Signal Frame Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
8:0	SIGFRAME	0x000	RW	**Signal Frame**
	When a frame matching SIGFRAME is detected by the receiver, SIGF interrupt flag is set.			

S.5.7 LEUARTn_RXDATAX - Receive Buffer Data Extended 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						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.		
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.		

Bit	Name	Reset	Access	Description
8:0	RXDATA	0x000	R	**RX Data**
	Use this register to access data read from the LEUART. Buffer is cleared on read access.			

S.5.8 LEUARTn_RXDATA - Receive Buffer 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																									0x00							
Access																									R							
Name																									RXDATA							

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

S.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.		
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.		
8:0	RXDATAP	0x000	R	**RX Data Peek**
	Use this register to access data read from the LEUART.			

S.5.10 LEUARTn_TXDATAX - Transmit Buffer Data Extended Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
15	RXENAT	0	W	**Enable RX After Transmission**

Set to enable reception after transmission.

Value	Description
0	-
1	The receiver is enabled, setting RXENS after the frame has been transmitted

14	TXDISAT	0	W	**Disable TX After Transmission**

Set to disable transmitter directly after transmission has competed.

Value	Description
0	-
1	The transmitter is disabled, clearing TXENS after the frame has been transmitted

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

12:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.

S.5.11 LEUARTn_TXDATA - Transmit Buffer Data Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
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.

S.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.		
10	SIGF	0	R	**Signal Frame Interrupt Flag**

Bit	Name	Reset	Access	Description
	Set when a signal frame is detected.			
9	STARTF	0	R	**Start Frame Interrupt Flag**
	Set when a start frame is detected.			
8	MPAF	0	R	**Multi-Processor Address Frame Interrupt Flag**
	Set when a multi-processor address frame is detected.			
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.			

S.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.		
10	SIGF	0	W1	**Set Signal Frame Interrupt Flag**
	Write to 1 to set the SIGF interrupt flag.			
9	STARTF	0	W1	**Set Start Frame Interrupt Flag**
	Write to 1 to set the STARTF interrupt flag.			
8	MPAF	0	W1	**Set Multi-Processor Address Frame Interrupt Flag**
	Write to 1 to set the MPAF interrupt flag.			
7	FERR	0	W1	**Set Framing Error Interrupt Flag**
	Write to 1 to set the FERR interrupt flag.			
6	PERR	0	W1	**Set Parity Error Interrupt Flag**
	Write to 1 to set the PERR interrupt flag.			
5	TXOF	0	W1	**Set TX Overflow Interrupt Flag**
	Write to 1 to set the TXOF interrupt flag.			
4	RXUF	0	W1	**Set RX Underflow Interrupt Flag**
	Write to 1 to set the RXUF interrupt flag.			
3	RXOF	0	W1	**Set RX Overflow Interrupt Flag**
	Write to 1 to set the RXOF interrupt flag.			
2:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	TXC	0	W1	**Set TX Complete Interrupt Flag**
	Write to 1 to set the TXC interrupt flag.			

S.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																						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.		
10	SIGF	0	W1	**Clear Signal-Frame Interrupt Flag**
	Write to 1 to clear the SIGF interrupt flag.			
9	STARTF	0	W1	**Clear Start-Frame Interrupt Flag**
	Write to 1 to clear the STARTF interrupt flag.			
8	MPAF	0	W1	**Clear Multi-Processor Address Frame Interrupt Flag**
	Write to 1 to clear the MPAF interrupt flag.			
7	FERR	0	W1	**Clear Framing Error Interrupt Flag**
	Write to 1 to clear the FERR interrupt flag.			
6	PERR	0	W1	**Clear Parity Error Interrupt Flag**
	Write to 1 to clear the PERR interrupt flag.			
5	TXOF	0	W1	**Clear TX Overflow Interrupt Flag**
	Write to 1 to clear the TXOF interrupt flag.			
4	RXUF	0	W1	**Clear RX Underflow Interrupt Flag**
	Write to 1 to clear the RXUF interrupt flag.			
3	RXOF	0	W1	**Clear RX Overflow Interrupt Flag**
	Write to 1 to clear the RXOF interrupt flag.			
2:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	TXC	0	W1	**Clear TX Complete Interrupt Flag**
	Write to 1 to clear the TXC interrupt flag.			

S.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.		
10	SIGF	0	RW	**Signal Frame Interrupt Enable**
	Enable interrupt on signal frame.			
9	STARTF	0	RW	**Start Frame Interrupt Enable**
	Enable interrupt on start frame.			
8	MPAF	0	RW	**Multi-Processor Address Frame Interrupt Enable**
	Enable interrupt on multi-processor address frame.			
7	FERR	0	RW	**Framing Error Interrupt Enable**

Bit	Name	Reset	Access	Description
	Enable interrupt on framing error.			
6	PERR	0	RW	**Parity Error Interrupt Enable**
	Enable interrupt on parity error.			
5	TXOF	0	RW	**TX Overflow Interrupt Enable**
	Enable interrupt on TX overflow.			
4	RXUF	0	RW	**RX Underflow Interrupt Enable**
	Enable interrupt on RX underflow.			
3	RXOF	0	RW	**RX Overflow Interrupt Enable**
	Enable interrupt on RX overflow.			
2	RXDATAV	0	RW	**RX Data Valid Interrupt Enable**
	Enable interrupt on RX data.			
1	TXBL	0	RW	**TX Buffer Level Interrupt Enable**
	Enable interrupt on TX buffer level.			
0	TXC	0	RW	**TX Complete Interrupt Enable**
	Enable interrupt on TX complete.			

S.5.16 LEUARTn_PULSECTRL - Pulse Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
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.			

S.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.		
0	REGFREEZE	0	RW	**Register Update Freeze**

When set, the update of the LEUART 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.

S.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.		
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.			

S.5.19 LEUARTn_ROUTE - I/O Routing Register

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

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	LOCATION	0x0	RW	**I/O Location**

Decides the location of the LEUART I/O pins.

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3
4	LOC4	Location 4

7:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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

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

S.5.20 LEUARTn_INPUT - LEUART Input 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	0x0			
Access																												RW	RW			
Name																												RXPRS	RXPRSSEL			

Bit	Name	Reset	Access	Description
31:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4	RXPRS	0	RW	**PRS RX Enable**

When set, the PRS channel selected as input to RX.

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

T ARM Timer/Counter

T.1 Introduction

The 16-bit general purpose Timer has 3 compare/capture channels for input capture and compare/Pulse-Width Modulation (PWM) output. TIMER0 also includes a Dead-Time Insertion module suitable for motor control applications.

T.2 Features

- 16-bit auto reload up/down counter
  - Dedicated 16-bit reload register which serves as counter maximum
- 3 Compare/Capture channels
  - Individual 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 (break)
- Interrupts, PRS output and/or DMA request
  - Underflow
  - Overflow
  - Compare/Capture event
- Dead-Time Insertion Unit (TIMER0 only)
  - 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

T.3 Functional Description

An overview of the TIMER module is shown in Figure 198. The Timer module consists of a 16 bit up/down counter with 3 Compare/Capture channels connected to pins TIMn_CC0, TIMn_CC1, and TIMn_CC2.

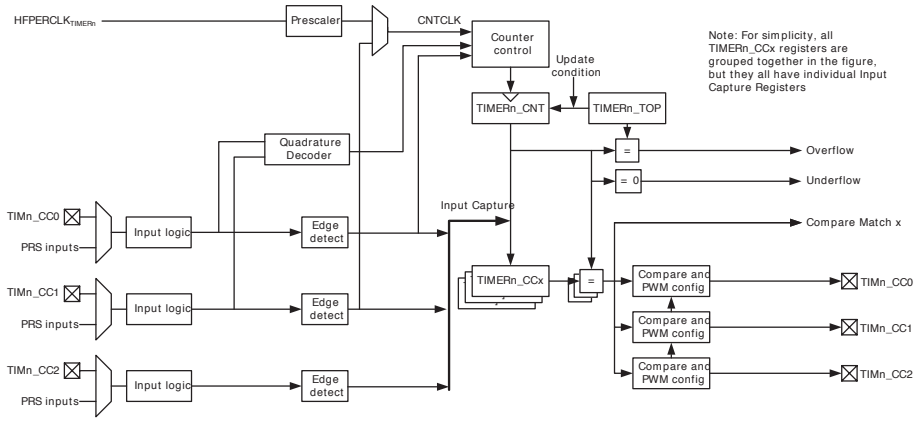


Figure 198: TIMER Block Overview

T.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 modes listed above, the TIMER 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.

Events

Overflow is set when the counter value shifts from TIMERn_TOP to the next value when counting up. In up-count 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, the next value is TIMERn_TOP. In up/down-count mode the next value is 1.

Update event is set on overflow in up-count mode and on underflow in down-count or up/down count mode. This event is used to time updates of buffered values.

Operation

Figure 199 shows the hardware Timer/Counter control. Software can start or stop the counter by writing a 1 to the START or STOP bits in TIMERn_CMD. The counter value (CNT in TIMERn_CNT) can always be written by software to any 16-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.

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.

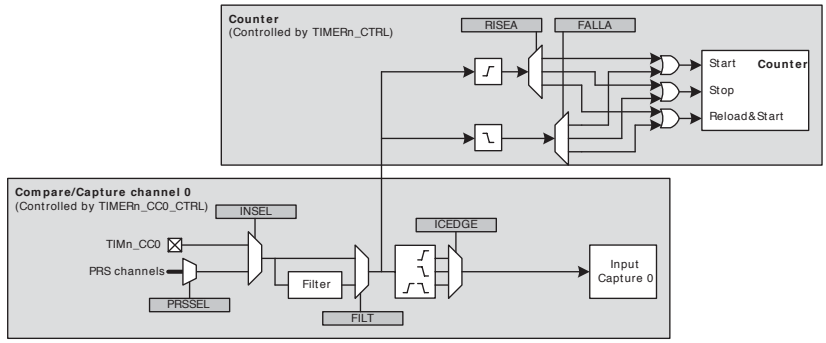


Figure 199: TIMER Hardware Timer/-Counter Control

Clock Source

The counter can be clocked from several sources, which are all synchronized with the peripheral clock (HFPERCLK). See Figure 200.

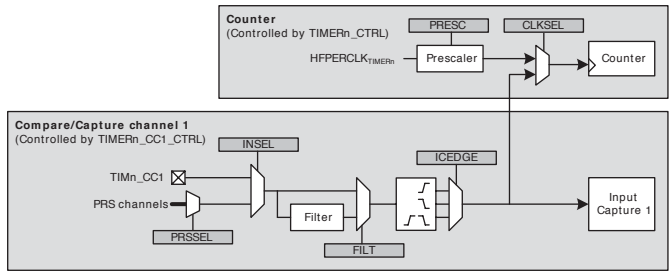


Figure 200: TIMER Clock Selection

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 result in incorrect result. The prescaler is stopped and reset when the timer is stopped.

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.

Underflow/Overflow from Neighboring Timer All Timers are linked together (see Figure 201), 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.

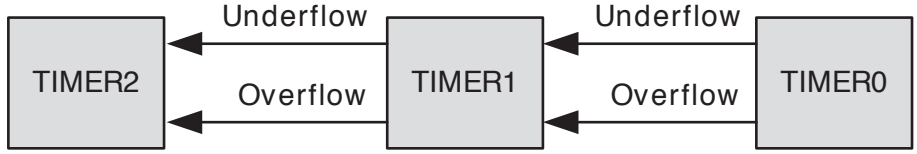


Figure 201: TIMER Connections

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*. 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.

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 have not yet been written to the TIMERn_TOP register (see Figure 202.

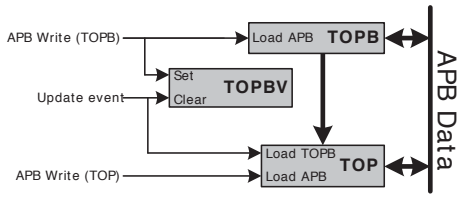


Figure 202: TIMER TOP Value Update Functionality

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 203).

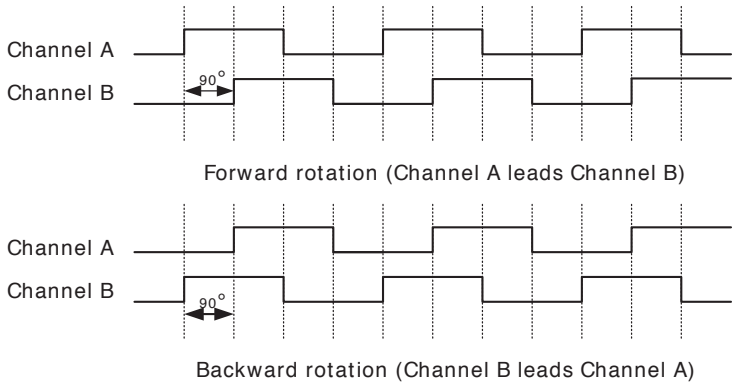


Figure 203: TIMER Quadrature Encoded Inputs

In the Timer these inputs are tapped from the Compare/Capture channel 0 (Channel A) and 1 (Channel B) inputs before edge detection. The Timer/Counter then increments or decrements the counter, based on the phase relation between the two inputs. The 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.

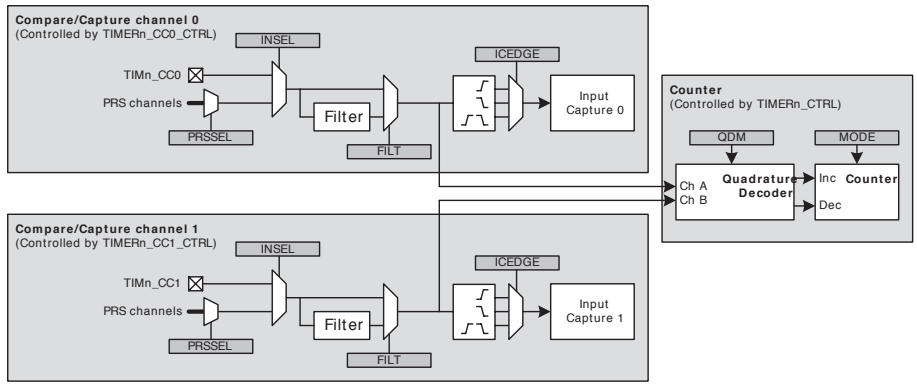


Figure 204: TIMER Quadrature Decoder Configuration

The Quadrature Decoder can be set in either X2 or X4 mode, which is configured in the QDM bit in TIMERn_CTRL. See Figure 204

X2 Decoding Mode In X2 Decoding mode, the counter increments or decrements on every edge of Channel A, see Table 205 and Figure 206.

Figure 205: TIMER Counter Response in X2 Decoding Mode

Channel B	Channel A	
	Rising	Falling
0	Increment	Decrement
1	Decrement	Increment

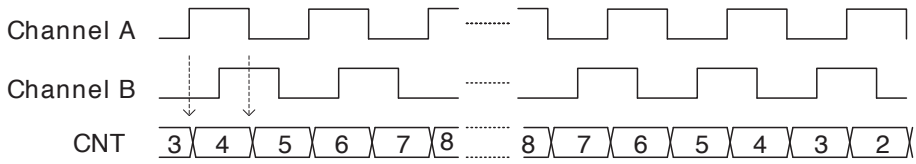


Figure 206: TIMER X2 Decoding Mode

X4 Decoding Mode In X4 Decoding mode, the counter increments or decrements on every edge of Channel A and Channel B, see Figure 208 and Table 207.

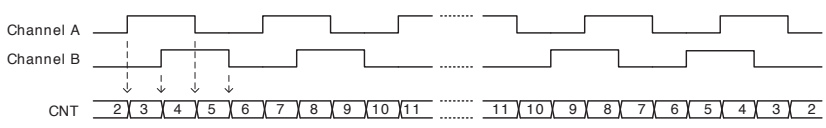


Figure 208: TIMER X4 Decoding Mode

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 207: TIMER Counter Response in X4 Decoding Mode

TIMER Rotational Position To calculate a position Equation T.3.1 can be used.

$$pos° = (CNT/X \times N) \times 360° \quad (16)$$

where X = Encoding type and N = Number of pulses per revolution.

T.3.2 Compare/Capture Channels

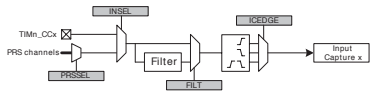
The Timer contains 3 Compare/Capture channels, which can be configured in the following modes:

1. Input Capture
2. Output Compare
3. PWM

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 209). 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 209: TIMER Input Pin Logic



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.

Input Capture mode When running in Input Capture mode, TIMERn_CCx_CCV and TIMERn_CCx_CCVB form a FIFO buffer, and new capture values are added on a capture event, see Figure 210. 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 case a capture is triggered while both CCV and CCVB contain unread capture values, the buffer overflow interrupt flag (ICBOF in TIMERn_IF) will be set. New capture values will on overflow overwrite the value in TIMERn_CCx_CCVB.

In input capture mode, the timer will only trigger interrupts when it is running

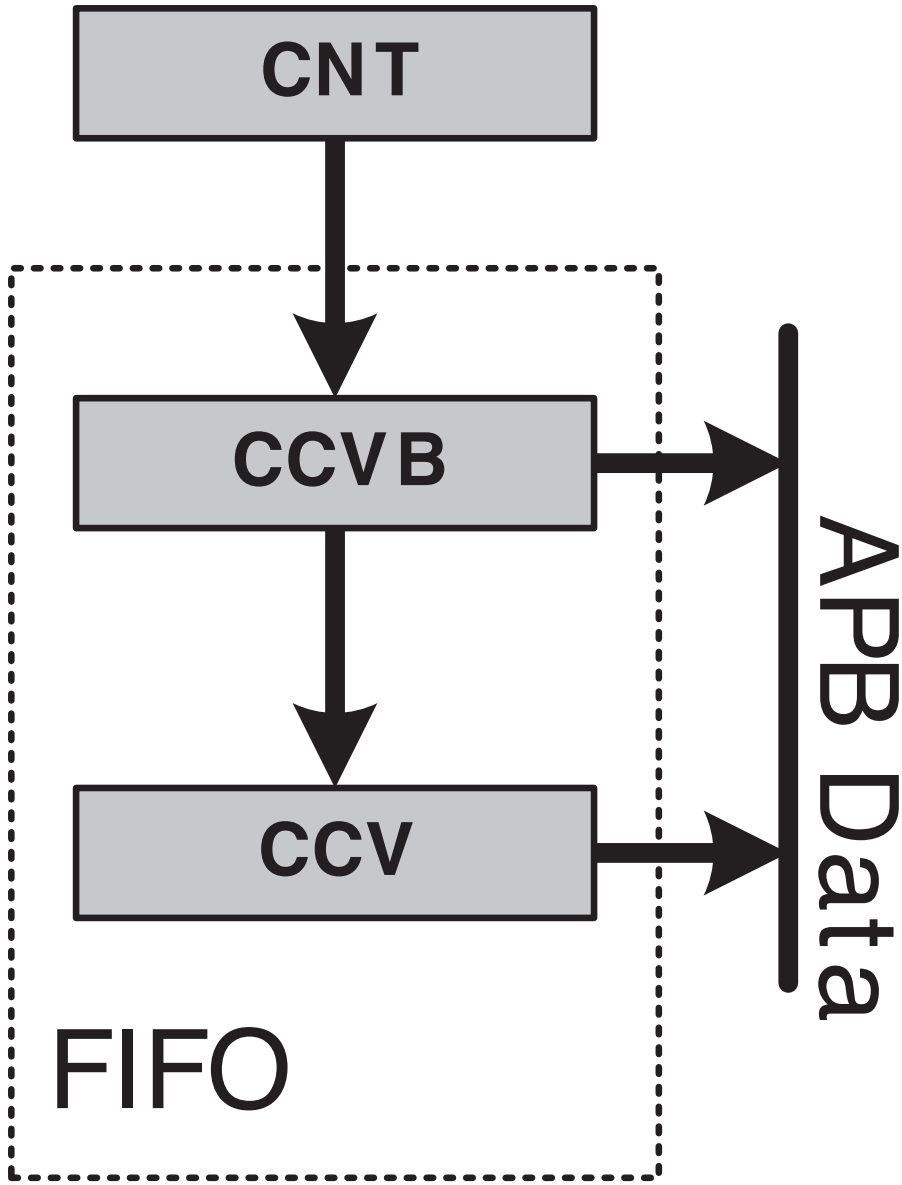


Figure 210: TIMER Input Capture Buffer Functionality

Compare and PWM Mode 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 211. 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 have not yet been written to the TIMERn_CCx_CCV register. Note that when writing 0 to TIMERn_CCx_CCVB 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.

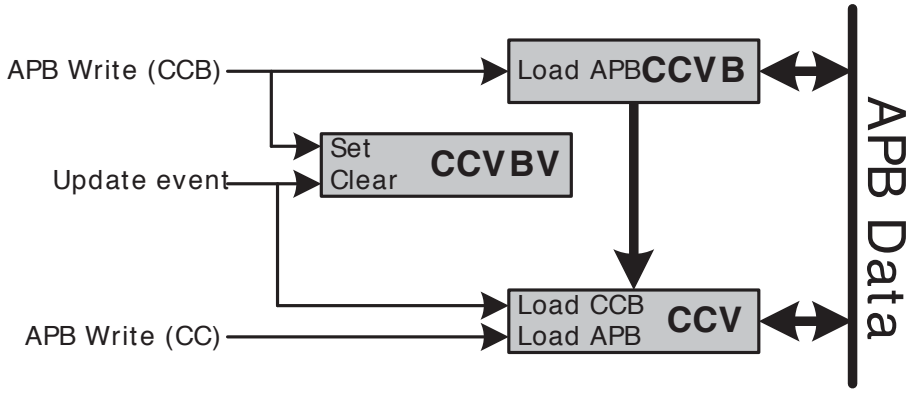


Figure 211: TIMER Output Compare/PWM Buffer Functionality

Input Capture

In Input Capture Mode, the counter value (TIMERn_CNT) can be captured in the Compare/Capture Register (TIMERn_CCx_CCV), see Figure 212. In this mode, TIMERn_CCx_CCV is read-only. Together with the Compare/Capture Buffer Register (TIMERn_CCx_CCVB) the TIMERn_CCx_CCV form a double-buffered capture registers allowing two subsequent capture events to take place before a read-out is required. The CCPOL bits in TIMERn_STATUS indicate the polarity the edge that triggered the capture in TIMERn_CCx_CCV.

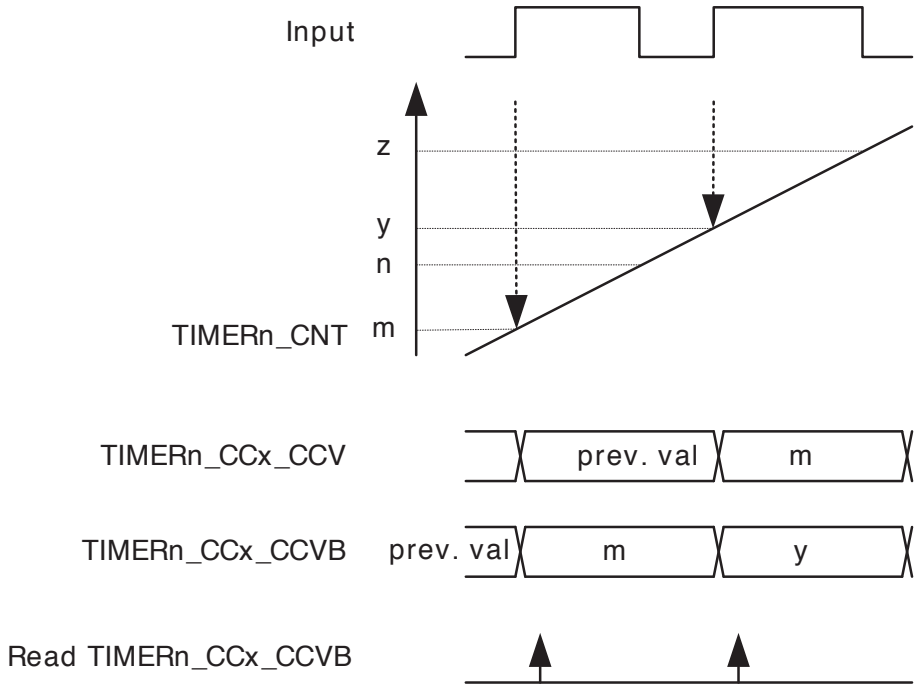


Figure 212: TIMER Input Capture

Period/Pulse-Width Capture Period and/or pulse-width capture can be achieved by setting the RISEA field in TIMERn_CTRL to Clear&Start, and select the wanted input from either external pin or PRS, see Figure 213. 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 a the width of a high pulse, the Compare/Capture Channel should be set to capture on a falling edge of the input signal. To start the measuring period on either a falling edge or measure the low pulse-width of a signal, opposite polarities should be chosen.

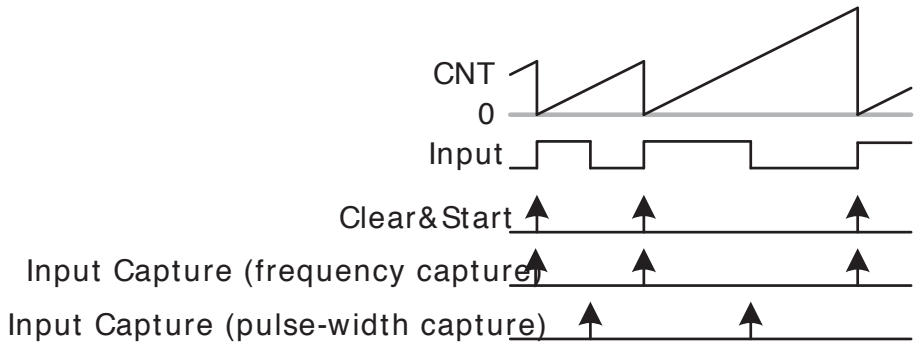


Figure 213: TIMER Period and/or Pulse width Capture

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 214. 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.

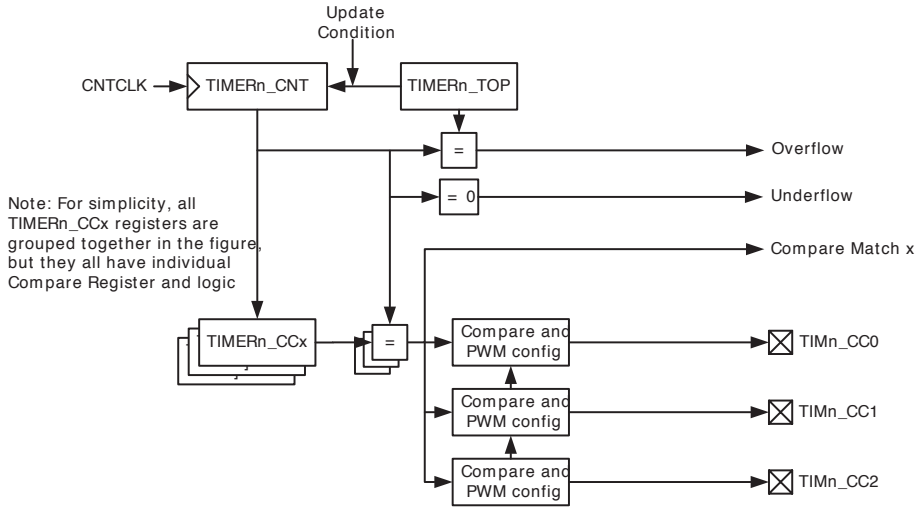


Figure 214: TIMER Block Diagram Showing Comparison Functionality

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.

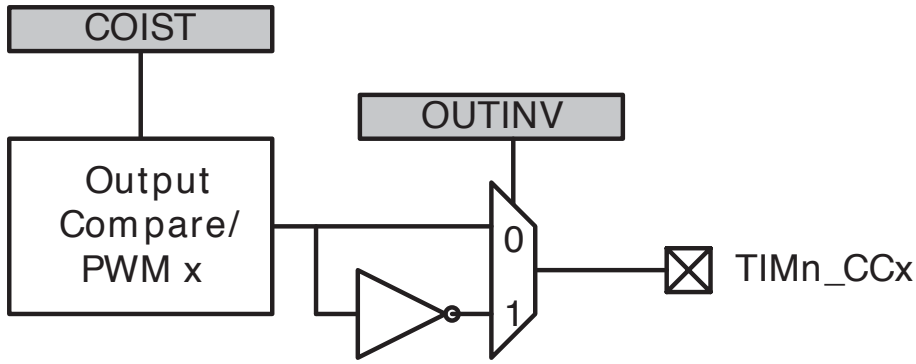


Figure 215: TIMER Output Logic

Frequency Generation (FRG) Frequency generation (see Figure 216) 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

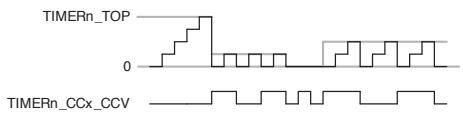


Figure 216: TIMER Up-count Frequency Generation

The output frequency is given by Equation T.3.2

$$f_{FRG} = f_{HFPERCLK} / (2^{(PRESC + 1)} \times (TOP + 1) \times 2) \quad (17)$$

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.

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 217). 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 TIMER_CCx to TOP+1 or higher. The PWM resolution (in bits) is then given by Equation T.3.2.

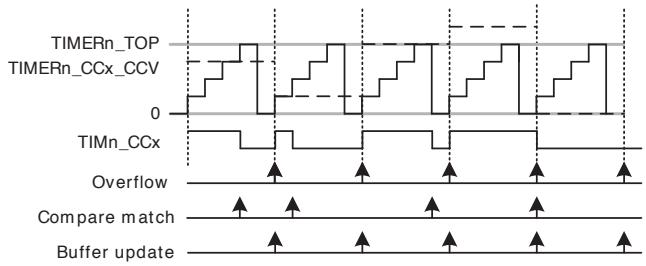


Figure 217: TIMER Up-count PWM Generation

$$R_{PWM_{up}} = \log(TOP+1)/\log(2) \quad (18)$$

The PWM frequency is given by Equation T.3.2:

$$f_{PWM_{up/down}} = f_{HFPERCLK}/(\ 2^{PRESC} \times (TOP + 1) \quad (19)$$

The high duty cycle is given by Equation T.3.2

$$DS_{up} = CCVx/TOP \quad (20)$$

2x Count Mode When the Timer is set in 2x mode, the TIMER 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 218

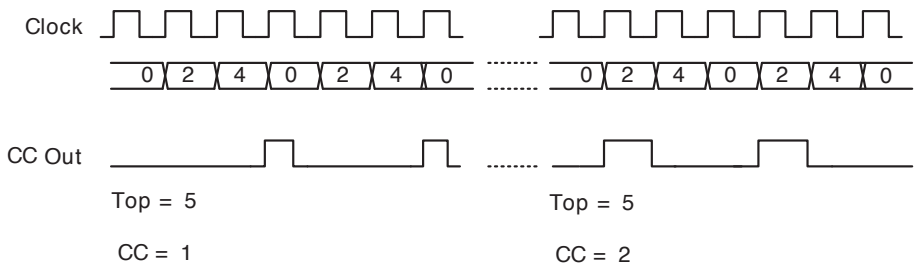


Figure 218: TIMER CC out in 2x mode

The mode is enabled by setting the X2CNT field in TIMERn_CTRL register. The intended use of the 2x mode is to generate 2x PWM frequency when the Compare/-Capture channel is put in PWM mode. Since the PWM output is updated on both edges of the clock, frequency prescaling will result in incorrect result in this mode. The PWM resolution (in bits) is then given by Equation T.3.2.

$$R_{PWM_{2xmode}} = \log(TOP/2+1)/\log(2) \quad (21)$$

The PWM frequency is given by Equation T.3.2:

$$f_{PWM_{2xmode}} = 2 \times f_{HFPERCLK} / \text{floor}(TOP/2)+1 \quad (22)$$

The high duty cycle is given by Equation T.3.2

$$DS_{2xmode} = CCVx/TOP \quad (23)$$

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 219.The resolution (in bits) is given by Equation T.3.2.

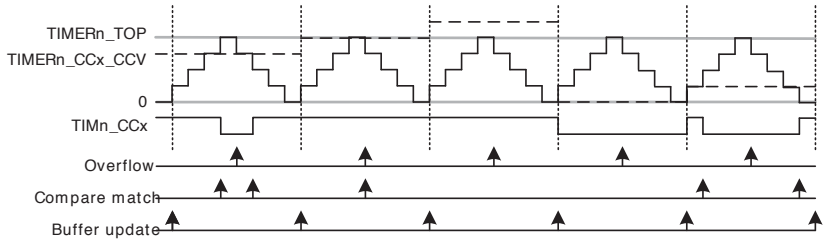


Figure 219: TIMER Up/Down-count PWM Generation

$R_{PWM_{up/down}} = \log(TOP+1)/\log(2)$ (24)

The PWM frequency is given by Equation T.3.2:

$f_{PWM_{up/down}} = f_{HFPERCLK}/(2^{(PRESC+1)} \times TOP)$ (25)

The high duty cycle is given by Equation T.3.2

$DS_{up/down} = CCVx/TOP$ (26)

2x Count Mode When the Timer is set in 2x mode, the TIMER will count up/down by two. This will in effect make any odd Top value be rounded down to the closest even number. Similarly, any odd CC value will generate a match on the closest lower even value as shown in Figure 220

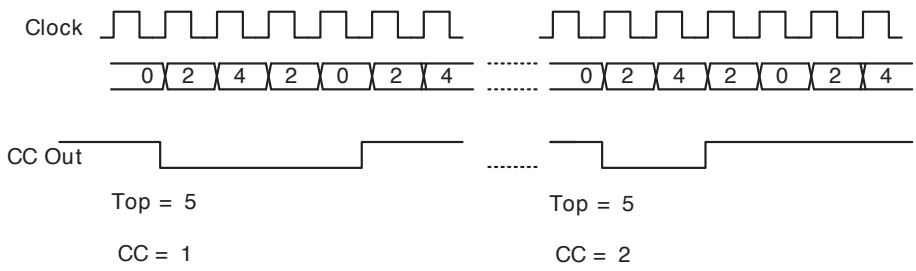


Figure 220: TIMER CC out in 2x mode

The mode is enabled by setting the X2CNT field in TIMERn_CTRL register. The intended use of the 2x mode is to generate 2x PWM frequency when the Compare/-Capture channel is put in PWM mode. Since the PWM output is updated on both edges of the clock, frequency prescaling will result in incorrect result in this mode. The PWM resolution (in bits) is then given by Equation T.3.2.

$$R_{PWM_{2xmode}} = \log(TOP/2+1)/\log(2) \quad (27)$$

The PWM frequency is given by Equation T.3.2:

$$f_{PWM_{2xmode}} = f_{HFPERCLK}/TOP \quad (28)$$

The high duty cycle is given by Equation T.3.2

$$DS_{2xmode} = CCVx/TOP \quad (29)$$

T.3.3 Dead-Time Insertion Unit (TIMER0 only)

The Dead-Time Insertion Unit aims to make control of BLDC motors safer and more efficient by introducing complementary PWM outputs with dead-time insertion and fault handling, see Figure 221.

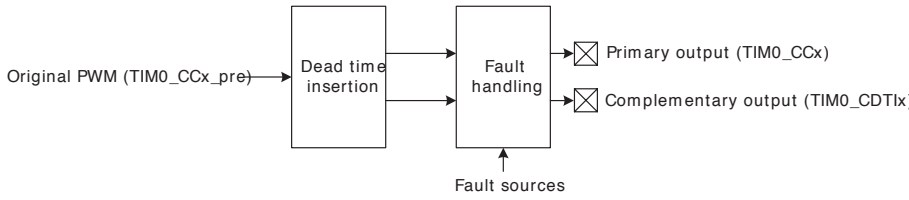


Figure 221: TIMER Dead-Time Insertion Unit Overview

When used for motor control, the PWM outputs TIM0_CC0, TIM0_CC1 and TIM0_CC2 are often connected to the high-side transistors of a triple half-bridge setup (UH, VH and WH), and the complementary outputs connected to the respective low-side transistors (UL, VL, WL shown in Figure 222). 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.

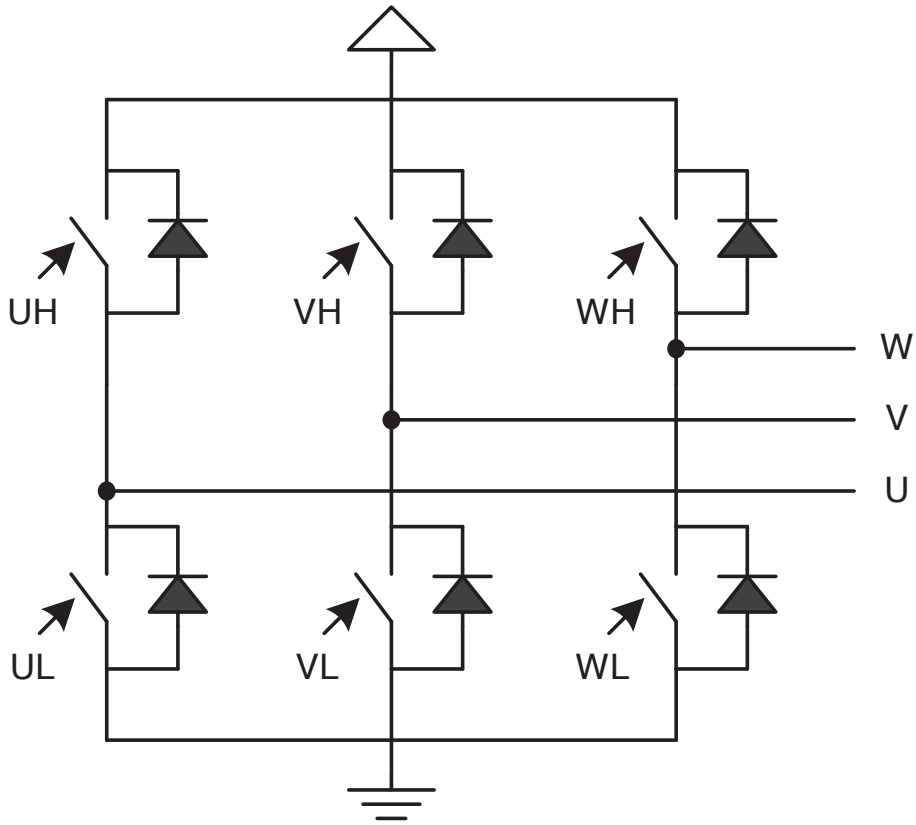


Figure 222: TIMER Triple Half-Bridge

For each of the 3 compare-match outputs of TIMER0, an additional complementary output is provided by the DTI unit. These outputs, named TIM0_CDTI0, TIM0_CDTI1 and TIM0_CDTI2 are provided to make control of e.g. 3-channel BLDC or PMAC motors possible using only a single timer, see Figure 223.

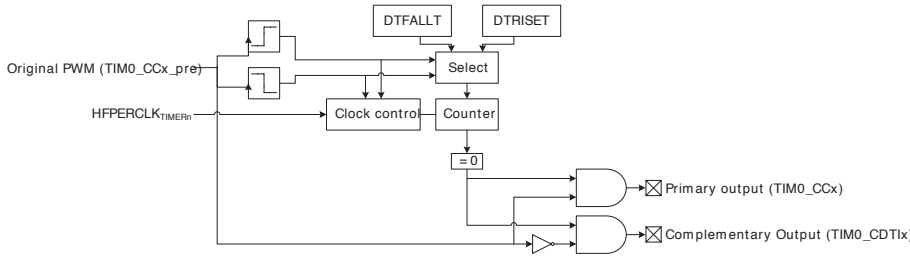


Figure 223: TIMER Overview of Dead-Time Insertion Block for a Single PWM channel

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 224.

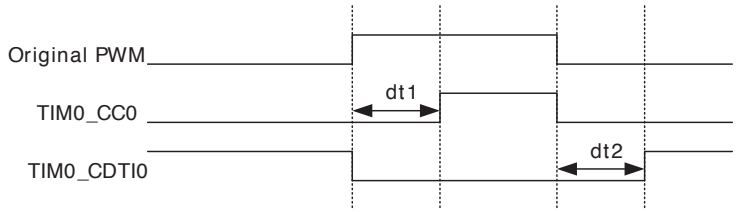


Figure 224: TIMER Polarity of Both Signals are Set as Active-High

Dead-time is specified individually for the rising and falling edge of the original PWM. These values are shared across all the three PWM channels of the DTI unit. A single prescaler value is provided for the DTI unit, meaning that both the rising and falling edge dead-times share prescaler value. The prescaler divides the $HFPERCLK_{TIMERn}$ 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.

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 however, if this is required by the application. The active values of the primary and complementary outputs are set by two 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 225

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.

exampleTIMER DTI Example 1

DTIPOL = 0 and DTCINV = 0 results in outputs with opposite phase and active-high states.

exampleTIMER DTI Example 2

DTIPOL = 1 and DTCINV = 1 results in outputs with equal phase. The primary output will be active-high, while the complementary will be active-low

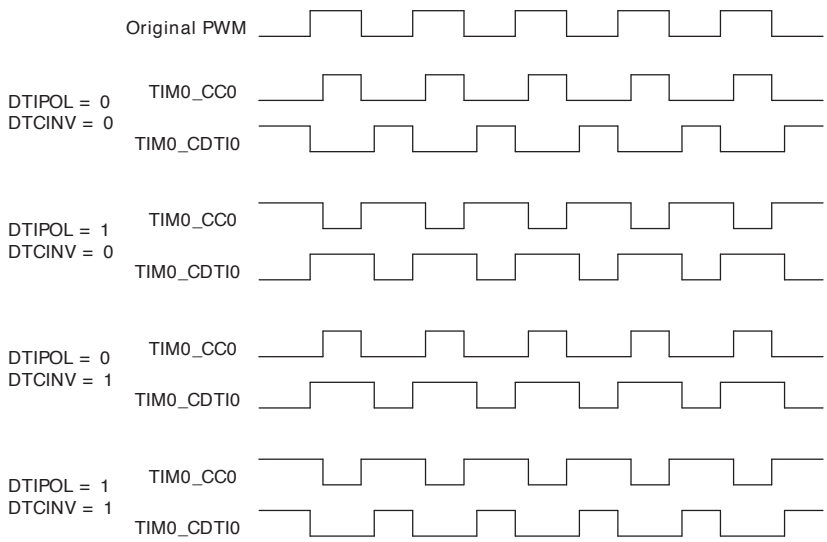


Figure 225: TIMER Output Polarities

Output generation on the individual DTI outputs can be disabled by configuring TIMER0_DTOGEN. When output generation on an output is disabled, it will go to and stay in its inactive state.

PRS Channel as Source

A PRS channel can optionally be used as input to the DTI module instead of the PWM output from the timer. 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 PRS channel to use is chosen by configuring DTPRSSEL in TIMER0_DTCTRL. Note that the timer must be running even when PRS is used as DTI source.

The DTI prescaler, set by DTPRESC in TIMER0_DTTIME determines with which accuracy 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.

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 makes a fast reaction to faults possible, reducing the possibility of damage to the system.

The fault sources which trigger a fault in the DTI module are determined by TIMER0_DTFSEN. 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. Please note that for Core Lockup, the LOCKUPRDIS in RMU_CTRL must be set. Otherwise this will generate a full reset of the EFM32.

Action on Fault When a fault occurs, the bit representing the fault source is set in DTFS, 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 of TIMER0_DTFS. TIMER0_DTFS is organized in the same way as DTFSEN, with one bit for each source.

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 TIMER0_DTFS. If the fault cause, determined by TIMER0_DTFS, is the debugger alone, the outputs can optionally be re-enabled when the debugger exits and the processor resumes normal operation. The corresponding bit in TIMER0_DTFS will in that case be cleared by hardware. The automatic start-up functionality can be enabled by setting DTDAS in TIMER0_DTCTRL. If more bits are still set in DTFS when the automatic start-up functionality has cleared the debugger bit, the DTI module does not exit the fault state. The fault state is only exited when all the bits in TIMER0_DTFS have been cleared.

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 for 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.

T.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 DBGHALT in TIMERn_CTRL.

T.3.5 Interrupts, DMA and PRS Output

The Timer has 5 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.

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 226. 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.

Event	Acknowledge
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

Figure 226: TIMER Events

T.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.

T.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_IEN	RW	Interrupt Enable Register
0x010	TIMERn_IF	R	Interrupt Flag Register
0x014	TIMERn_IFS	W1	Interrupt Flag Set Register
0x018	TIMERn_IFC	W1	Interrupt Flag Clear 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
0x028	TIMERn_ROUTE	RW	I/O Routing Register
0x030	TIMERn_CC0_CTRL	RW	CC Channel Control Register
0x034	TIMERn_CC0_CCV	RWH	CC Channel Value Register
0x038	TIMERn_CC0_CCVP	R	CC Channel Value Peek Register
0x03C	TIMERn_CC0_CCVB	RWH	CC Channel Buffer Register
0x040	TIMERn_CC1_CTRL	RW	CC Channel Control Register
0x044	TIMERn_CC1_CCV	RWH	CC Channel Value Register
0x048	TIMERn_CC1_CCVP	R	CC Channel Value Peek Register
0x04C	TIMERn_CC1_CCVB	RWH	CC Channel Buffer Register
0x050	TIMERn_CC2_CTRL	RW	CC Channel Control Register
0x054	TIMERn_CC2_CCV	RWH	CC Channel Value Register
0x058	TIMERn_CC2_CCVP	R	CC Channel Value Peek Register
0x05C	TIMERn_CC2_CCVB	RWH	CC Channel Buffer Register
0x070	TIMERn_DTCTRL	RW	DTI Control Register
0x074	TIMERn_DTTIME	RW	DTI Time Control Register
0x078	TIMERn_DTFC	RW	DTI Fault Configuration Register
0x07C	TIMERn_DTOGEN	RW	DTI Output Generation Enable Register
0x080	TIMERn_DTFAULT	R	DTI Fault Register
0x084	TIMERn_DTFAULTC	W1	DTI Fault Clear Register

Offset	Name	Type	Description
0x088	TIMERn_DTLOCK	RW	DTI Configuration Lock Register

T.5 Register Description

T.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		0x0		0x0		0	0	0	0	0		0x0	
Access			RW	RW	RW										RW				RW		RW		RW		RW	RW	RW	RW	RW		RW	
Name			RSSCOIST	ATI	PRESC										CLKSEL				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.		
29	RSSCOIST	0	RW	**Reload-Start Sets Compare Ouptut initial State**
	When enabled, compare output is set to COIST value at Reload-Start event			
28	ATI	0	RW	**Always Track Inputs**
	Enable ATI 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.		
17:16	CLKSEL	0x0	RW	**Clock Source Select**
	These bits select the clock source for the timer.			

Value	Mode	Description
0	PRESCHFPERCLK	Prescaled HFPERCLK
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

15:14	Reserved	To ensure compatibility with future devices, always write bits to 0.		
13	X2CNT	0	RW	**2x Count Mode**
	Enable 2x count mode			
12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

Bit	Name	Reset	Access	Description

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

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.

1:0 MODE 0x0 RW **Timer Mode**

These bit 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

T.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

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	STOP	0	W1	**Stop Timer**
	Write a 1 to this bit to stop timer			
0	START	0	W1	**Start Timer**
	Write a 1 to this bit to start timer			

T.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
Access						R	R	R						R	R	R						R	R	R						R	R	R
Name						CCPOL2	CCPOL1	CCPOL0						ICV2	ICV1	ICV0						CCVBV2	CCVBV1	CCVBV0						TOPBV	DIR	RUNNING

Bit	Name	Reset	Access	Description
31:27	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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

23:19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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).

Bit	Name	Reset	Access	Description
	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)		
15:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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		
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		
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		
7:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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		
1	DIR	0	R	**Direction**
	Indicates count direction.			

Value	Mode	Description
0	UP	Counting up
1	DOWN	Counting down

0	RUNNING	0	R	**Running**
	Indicates if timer is running or not.			

T.5.4 TIMERn_IEN - Interrupt Enable 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
Access																						RW	RW	RW		RW	RW	RW			RW	RW
Name																						ICBOF2	ICBOF1	ICBOF0		CC2	CC1	CC0			UF	OF

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10	ICBOF2	0	RW	**CC Channel 2 Input Capture Buffer Overflow Interrupt Enable**
	Enable/disable Compare/Capture ch 2 input capture buffer overflow interrupt.			
9	ICBOF1	0	RW	**CC Channel 1 Input Capture Buffer Overflow Interrupt Enable**
	Enable/disable Compare/Capture ch 1 input capture buffer overflow interrupt.			

Bit	Name	Reset	Access	Description
8	ICBOF0	0	RW	**CC Channel 0 Input Capture Buffer Overflow Interrupt Enable**
	Enable/disable Compare/Capture ch 0 input capture buffer overflow interrupt.			
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6	CC2	0	RW	**CC Channel 2 Interrupt Enable**
	Enable/disable Compare/Capture ch 2 interrupt.			
5	CC1	0	RW	**CC Channel 1 Interrupt Enable**
	Enable/disable Compare/Capture ch 1 interrupt.			
4	CC0	0	RW	**CC Channel 0 Interrupt Enable**
	Enable/disable Compare/Capture ch 0 interrupt.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	UF	0	RW	**Underflow Interrupt Enable**
	Enable/disable underflow interrupt.			
0	OF	0	RW	**Overflow Interrupt Enable**
	Enable/disable overflow interrupt.			

T.5.5 TIMERn_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	0	0		0	0	0			0	0
Access																						R	R	R		R	R	R			R	R
Name																						ICBOF2	ICBOF1	ICBOF0		CC2	CC1	CC0			UF	OF

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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 the TIMERn_CC2_CCV/TIMERn_CC2_CCVB register pair.			
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 the TIMERn_CC1_CCV/TIMERn_CC1_CCVB register pair.			
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 the TIMERn_CC0_CCV/TIMERn_CC0_CCVB register pair.			
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

T.5.6 TIMERn_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	0	0		0	0	0			0	0

Offset	Bit Position																											
Access																					W1	W1	W1		W1	W1	W1		W1	W1
Name																					ICBOF2	ICBOF1	ICBOF0		CC2	CC1	CC0		UF	OF

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10	ICBOF2	0	W1	**CC Channel 2 Input Capture Buffer Overflow Interrupt Flag Set**
	Writing a 1 to this bit will set Compare/Capture channel 2 input capture buffer overflow interrupt flag.			
9	ICBOF1	0	W1	**CC Channel 1 Input Capture Buffer Overflow Interrupt Flag Set**
	Writing a 1 to this bit will set Compare/Capture channel 1 input capture buffer overflow interrupt flag.			
8	ICBOF0	0	W1	**CC Channel 0 Input Capture Buffer Overflow Interrupt Flag Set**
	Writing a 1 to this bit will set Compare/Capture channel 0 input capture buffer overflow interrupt flag.			
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6	CC2	0	W1	**CC Channel 2 Interrupt Flag Set**
	Writing a 1 to this bit will set Compare/Capture channel 2 interrupt flag.			
5	CC1	0	W1	**CC Channel 1 Interrupt Flag Set**
	Writing a 1 to this bit will set Compare/Capture channel 1 interrupt flag.			
4	CC0	0	W1	**CC Channel 0 Interrupt Flag Set**
	Writing a 1 to this bit will set Compare/Capture channel 0 interrupt flag.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	UF	0	W1	**Underflow Interrupt Flag Set**
	Writing a 1 to this bit will set the underflow interrupt flag.			
0	OF	0	W1	**Overflow Interrupt Flag Set**
	Writing a 1 to this bit will set the overflow interrupt flag.			

T.5.7 TIMERn_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	0	0		0	0	0			0	0
Access																						W1	W1	W1		W1	W1	W1			W1	W1
Name																						ICBOF2	ICBOF1	ICBOF0		CC2	CC1	CC0			UF	OF

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10	ICBOF2	0	W1	**CC Channel 2 Input Capture Buffer Overflow Interrupt Flag Clear**
	Writing a 1 to this bit will clear Compare/Capture channel 2 input capture buffer overflow interrupt flag.			
9	ICBOF1	0	W1	**CC Channel 1 Input Capture Buffer Overflow Interrupt Flag Clear**
	Writing a 1 to this bit will clear Compare/Capture channel 1 input capture buffer overflow interrupt flag.			
8	ICBOF0	0	W1	**CC Channel 0 Input Capture Buffer Overflow Interrupt Flag Clear**
	Writing a 1 to this bit will clear Compare/Capture channel 0 input capture buffer overflow interrupt flag.			
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6	CC2	0	W1	**CC Channel 2 Interrupt Flag Clear**
	Writing a 1 to this bit will clear Compare/Capture interrupt flag 2.			
5	CC1	0	W1	**CC Channel 1 Interrupt Flag Clear**
	Writing a 1 to this bit will clear Compare/Capture interrupt flag 1.			
4	CC0	0	W1	**CC Channel 0 Interrupt Flag Clear**
	Writing a 1 to this bit will clear Compare/Capture interrupt flag 0.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		

Bit	Name	Reset	Access	Description
1	UF	0	W1	**Underflow Interrupt Flag Clear**
	Writing a 1 to this bit will clear the underflow interrupt flag.			
0	OF	0	W1	**Overflow Interrupt Flag Clear**
	Writing a 1 to this bit will clear th overflow interrupt flag.			

T.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																	0xFFFF															
Access																	RWH															
Name																	TOP															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	TOP	0xFFFF	RWH	**Counter Top Value**
	These bits hold the TOP value for the counter.			

T.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																	0x0000															
Access																	RW															
Name																	TOPB															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	TOPB	0x0000	RW	**Counter Top Value Buffer**
	These bits hold the TOP buffer value.			

T.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																	0x0000															
Access																	RWH															
Name																	CNT															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	CNT	0x0000	RWH	**Counter Value**
	These bits hold the counter value.			

T.5.11 TIMERn_ROUTE - I/O Routing Register

Offset	Bit Position																													
0x028	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset														0x0								0	0	0						0	0	0
Access														RW								RW	RW	RW						RW	RW	RW
Name														LOCATION								CDTI2PEN	CDTI1PEN	CDTI0PEN						CC2PEN	CC1PEN	CC0PEN

Bit	Name	Reset	Access	Description
31:19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18:16	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the CC and CDTI 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

15:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			
7:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

T.5.12 TIMERn_CCx_CTRL - CC Channel Control Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0		0x0				0	0	0x0						0x0		0x0		0x0					0		0	0x0	
Access					RW		RW				RW	RW	RW						RW		RW		RW					RW		RW	RW	
Name					ICEVCTRL		ICEDGE				FILT	INSEL	PRSSEL						CUFOA		COFOA		CMOA					COIST		OUTINV	MODE	

Bit	Name	Reset	Access	Description
31:28	Reserved	To ensure compatibility with future devices, always write bits to 0.		
27:26	ICEVCTRL	0x0	RW	**Input Capture Event Control**
	These bits control when a Compare/Capture PRS output pulse, interrupt flag and DMA request is set.			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	EVERYEDGE	PRS output pulse, interrupt flag and DMA request set on every capture
1	EVERYSECONDEDGE	PRS output pulse, interrupt flag and DMA request set on every second capture
2	RISING	PRS output pulse, interrupt flag and DMA request set on rising edge only (if ICEDGE = BOTH)
3	FALLING	PRS output pulse, interrupt flag and DMA request set on falling edge only (if ICEDGE = BOTH)

25:24 ICEDGE 0x0 RW **Input Capture Edge Select**

These bits control which edges the edge detector triggers on. The output is used for input capture and external clock input.

Value	Mode	Description
0	RISING	Rising edges detected
1	FALLING	Falling edges detected
2	BOTH	Both edges detected
3	NONE	No edge detection, signal is left as it is

23:22 Reserved To ensure compatibility with future devices, always write bits to 0.

21 FILT 0 RW **Digital Filter**

Enable digital filter.

Value	Mode	Description
0	DISABLE	Digital filter disabled
1	ENABLE	Digital filter enabled

20 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

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

15:14 Reserved To ensure compatibility with future devices, always write bits to 0.

13:12 CUFOA 0x0 RW **Counter Underflow Output Action**

Select output action on counter underflow.

Value	Mode	Description
0	NONE	No action on counter underflow
1	TOGGLE	Toggle output on counter underflow
2	CLEAR	Clear output on counter underflow
3	SET	Set output on counter underflow

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

Bit	Name	Reset	Access	Description
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

Bit	Name	Reset	Access	Description
7:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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 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. In PWM mode, the output will always be low when disabled, regardless of this bit. However, this bit will represent the initial value of the output, once it is enabled.

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	OUTINV	0	RW	**Output Invert**

Setting this bit inverts the output from the CC channel (Output compare,PWM).

Bit	Name	Reset	Access	Description
1:0	MODE	0x0	RW	**CC Channel Mode**

These bits select the mode for Compare/Capture channel.

Value	Mode	Description
0	OFF	Compare/Capture channel turned off
1	INPUTCAPTURE	Input capture
2	OUTPUTCOMPARE	Output compare
3	PWM	Pulse-Width Modulation

T.5.13 TIMERn_CCx_CCV - CC Channel Value Register

Offset	Bit Position																															
0x034	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0000															
Access																	RWH															
Name																	CCV															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	CCV	0x0000	RWH	**CC Channel Value**

In input capture mode, this field holds the first unread capture value. When reading this register in input capture mode, then 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.

T.5.14 TIMERn_CCx_CCVP - CC Channel Value Peek Register

Offset	Bit Position																															
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0000															
Access																	R															
Name																	CCVP															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	CCVP	0x0000	R	**CC Channel Value Peek**

This field is used to read the CC value without pulling data through the FIFO in capture mode.

T.5.15 TIMERn_CCx_CCVB - CC Channel Buffer 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																	0x0000															
Access																	RWH															
Name																	CCVB															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	CCVB	0x0000	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.			

T.5.16 TIMERn_DTCTRL - DTI 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								0																	0x0				0	0	0	0
Access								RW																	RW				RW	RW	RW	RW
Name								DTPRSEN																	DTPRSSEL				DTCINV	DTIPOL	DTDAS	DTEN

Bit	Name	Reset	Access	Description
31:25	Reserved	To ensure compatibility with future devices, always write bits to 0.		
24	DTPRSEN	0	RW	**DTI PRS Source Enable**
	Enable/disable PRS as DTI input.			
23:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:4	DTPRSSEL	0x0	RW	**DTI PRS Source Channel Select**
	Select which PRS channel to 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

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.			

Bit	Name	Reset	Access	Description
	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.			

T.5.17 TIMERn_DTTIME - DTI Time Control Register

Offset	Bit Position																													
0x074	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											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.		
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		
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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		
7:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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

T.5.18 TIMERn_DTFC - DTI Fault Configuration 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	0	0	0							0x0							0x0								0x0		
Access					RW	RW	RW	RW							RW							RW								RW		

Offset	Bit Position																													
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.		
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.		
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

15:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	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
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

7:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	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 0
2	PRSCH2	PRS Channel 2 selected as fault source 0
3	PRSCH3	PRS Channel 3 selected as fault source 0
4	PRSCH4	PRS Channel 4 selected as fault source 0
5	PRSCH5	PRS Channel 5 selected as fault source 0
6	PRSCH6	PRS Channel 6 selected as fault source 0
7	PRSCH7	PRS Channel 7 selected as fault source 0

T.5.19 TIMERn_DTOGEN - DTI Output Generation Enable 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

Offset	Bit Position																													
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.		
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.			

T.5.20 TIMERn_DTFAULT - DTI Fault 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
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.		
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.			

T.5.21 TIMERn_DTFAULTC - DTI Fault Clear 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

Offset	Bit Position																													
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.		
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.			

T.5.22 TIMERn_DTLOCK - DTI Configuration Lock 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																	0x0000															
Access																	RW															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	LOCKKEY	0x0000	RW	**DTI Lock Key**
	Write any other value than the unlock code to lock TIMER0_ROUTE, TIMER0_DTCTRL, TIMER0_DTTIME and TIMER0_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

U ARM Real Time Counter

U.1 Introduction

The Real Time Counter (RTC) contains a 24-bit counter and is clocked either by a 32.768 Hz crystal oscillator, a 32.768 Hz RC oscillator, or a 1 kHz RC oscillator . In addition to energy modes EM0 and EM1, the RTC is also available in EM2. This makes it ideal for keeping track of time since the RTC is enabled in EM2 where most of the device is powered down. Using the 1 kHz ULFRCO as input clock, the RTC can be used for timekeeping all the way down to EM3.

Two 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. They can also be used with the LETIMER to generate various output waveforms.

U.2 Features

- 24-bit Real Time Counter.
- Prescaler
  - 32.768 kHz/2^N, N = 0 - 15.
  - Overflow @ 0.14 hours for prescaler setting = 0.
  - Overflow @ 4660 hours (194 days) for prescaler setting = 15 (1 s tick).
- Two compare registers
  - A compare match can potentially wake-up the device from low energy modes EM1 and EM2.
  - Second compare register can be top value for RTC.
  - Both compare channels can trigger LETIMER.
  - Compare match events are available to other peripherals through the Peripheral Reflex System (PRS).

U.3 Functional Description

The RTC is a 24-bit counter with two compare channels. The RTC is closely coupled with the LETIMER, and can be configured to trigger it on a compare match on one or both compare channels. An overview of the RTC module is shown in Figure 227.

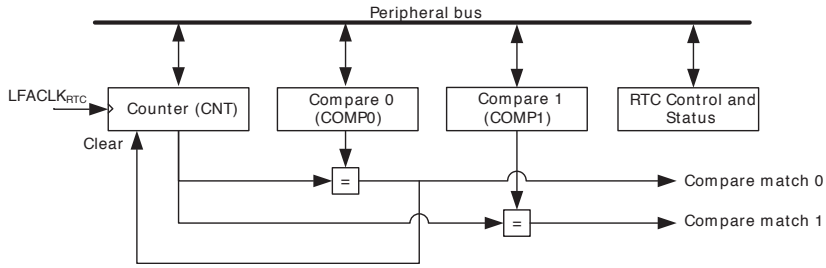


Figure 227: RTC Overview

U.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 on an overflow simply wrap around 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.

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 Equation U.3.1.

$$f_{RTC} = f_{LFACLK} / 2^{RTC\_PRESC} \quad (30)$$

where f_{LFACLK} is the LFACLK frequency (32.768 kHz) and RTC_PRESC is a 4 bit value. Table 228 shows the time of overflow and resolution of the RTC at the available prescaler values.

To use this module, the LE interface clock must be enabled in CMU_HFCORECLKEN0 in addition to the module clock

U.3.2 Compare Channels

Two compare channels are available in the RTC. The compare values can be set by writing to the RTC compare channel registers RTC_COMPn, 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).

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

Figure 228: RTC Resolution Vs Overflow

LETIMER Triggers

A compare event on either of the compare channels can start the LETIMER. See the LETIMER documentation for more information on this feature.

PRS Sources

Both 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.

U.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.

U.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.

U.3.5 Using the RTC in EM3

The RTC can be enabled all the way down to EM3 by using the ULFRCO as clock source. This is done by clearing CMU_LFCLKSEL_LFA and setting CMU_LFCLKSEL_LFAE to 1. This will make the RTC use the internal 1 kHz ultra low frequency RC oscillator (ULFRCO), consuming very little energy. Please note that the ULFRCO is not accurate over temperature and voltage, and it should be verified that the ULFRCO fulfills the timekeeping needs of the application before using this in the design.

U.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 [?].

U.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_COMP0	RW	Compare Value Register 0
0x00C	RTC_COMP1	RW	Compare Value Register 1
0x010	RTC_IF	R	Interrupt Flag Register
0x014	RTC_IFS	W1	Interrupt Flag Set Register
0x018	RTC_IFC	W1	Interrupt Flag Clear Register
0x01C	RTC_IEN	RW	Interrupt Enable Register
0x020	RTC_FREEZE	RW	Freeze Register
0x024	RTC_SYNCBUSY	R	Synchronization Busy Register

U.5 Register Description

U.5.1 RTC_CTRL - Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
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	
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		
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.			

U.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.		
23:0	CNT	0x000000	RWH	**Counter Value**
	Gives access to the counter value of the RTC.			

U.5.3 RTC_COMP0 - Compare Value Register 0 (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	0x000000																															
Access	RW																															
Name	COMP0																															

Bit	Name	Reset	Access	Description
31:24	Reserved	To ensure compatibility with future devices, always write bits to 0.		
23:0	COMP0	0x000000	RW	**Compare Value 0**
	A compare match event occurs when CNT is equal to this value. This event sets the COMP0 interrupt flag, and can be used to start the LETIMER. It is also available as a PRS signal.			

U.5.4 RTC_COMP1 - Compare Value Register 1 (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	0x000000																															
Access	RW																															
Name	COMP1																															

Bit	Name	Reset	Access	Description
31:24	Reserved	To ensure compatibility with future devices, always write bits to 0.		
23:0	COMP1	0x000000	RW	**Compare Value 1**
	A compare match event occurs when CNT is equal to this value. This event sets COMP1 interrupt flag, and can be used to start the LETIMER. It is also available as a PRS signal.			

U.5.5 RTC_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	0	0
Access																														R	R	R
Name																														COMP1	COMP0	OF

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	COMP1	0	R	**Compare Match 1 Interrupt Flag**
	Set on a compare match between CNT and COMP1.			
1	COMP0	0	R	**Compare Match 0 Interrupt Flag**
	Set on a compare match between CNT and COMP0.			
0	OF	0	R	**Overflow Interrupt Flag**
	Set on a CNT value overflow.			

U.5.6 RTC_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	0	0
Access																														W1	W1	W1
Name																														COMP1	COMP0	OF

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	COMP1	0	W1	**Set Compare match 1 Interrupt Flag**
	Write to 1 to set the COMP1 interrupt flag.			
1	COMP0	0	W1	**Set Compare match 0 Interrupt Flag**
	Write to 1 to set the COMP0 interrupt flag.			
0	OF	0	W1	**Set Overflow Interrupt Flag**
	Write to 1 to set the OF interrupt flag.			

U.5.7 RTC_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	0	0
Access																														W1	W1	W1
Name																														COMP1	COMP0	OF

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	COMP1	0	W1	**Clear Compare match 1 Interrupt Flag**
	Write to 1 to clear the COMP1 interrupt flag.			

Bit	Name	Reset	Access	Description
1	COMP0	0	W1	**Clear Compare match 0 Interrupt Flag**
	Write to 1 to clear the COMP0 interrupt flag.			
0	OF	0	W1	**Clear Overflow Interrupt Flag**
	Write to 1 to clear the OF interrupt flag.			

U.5.8 RTC_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	0	0
Access																														RW	RW	RW
Name																														COMP1	COMP0	OF

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	COMP1	0	RW	**Compare Match 1 Interrupt Enable**
	Enable interrupt on compare match 1.			
1	COMP0	0	RW	**Compare Match 0 Interrupt Enable**
	Enable interrupt on compare match 0.			
0	OF	0	RW	**Overflow Interrupt Enable**
	Enable interrupt on overflow.			

U.5.9 RTC_FREEZE - Freeze 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																																RW
Name																																REGFREEZE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	REGFREEZE	0	RW	**Register Update Freeze**
	When set, the update of the RTC 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 RTC register is updated into the Low Frequency domain as soon as possible.
1	FREEZE	The RTC is not updated with the new written value until the freeze bit is cleared.

U.5.10 RTC_SYNCBUSY - Synchronization Busy Register

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

Offset	Bit Position			
Reset		0	0	0
Access		R	R	R
Name		COMP1	COMP0	CTRL

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	COMP1	0	R	**COMP1 Register Busy**
	Set when the value written to COMP1 is being synchronized.			
1	COMP0	0	R	**COMP0 Register Busy**
	Set when the value written to COMP0 is being synchronized.			
0	CTRL	0	R	**CTRL Register Busy**
	Set when the value written to CTRL is being synchronized.			

V ARM Backup Real Time Counter

V.1 Introduction

The Backup Real Time Counter (BURTC) contains a 32-bit counter and is clocked either by a 32.768 kHz crystal oscillator, a 32.768 kHz RC oscillator, a 2kHz RC oscillator, or a 1kHz RC oscillator. A variety of prescaler settings are also available for the 32.768 kHz oscillators. The Backup RTC is available in all energy modes, making it ideal for time keeping with minimal energy consumption. The ability to keep running while the system is in backup mode allows the Backup RTC to keep track of time, even if the main power should drain out.

V.2 Features

- 32-bit Real Time Counter
- Prescaler for LFXO and LFRCO, 32.768 kHz/2^N, N = 0-7
- Available in all energy modes and backup mode.
- Timestamp and optionally switch to low power mode upon entry to backup mode.
- Oscillator failure detection.
- EM4 operation and wake-up.
- Not reset by system reset, only by software, pin reset, or power loss.
- Seamless frequency shifting while keeping track of time.
- 512 bytes of general purpose data retention.
- Detection of corrupt writes to retention registers when losing main power.
- PRS producer.

V.3 Functional Description

The Backup RTC is a 32-bit counter with one compare channel. The Backup RTC resides in a power domain which can be configured to always be on, in EM0 through EM4. This domain also has the possibility to be powered by a backup battery. For further details on the backup power domain, refer to [?]. Available in all energy modes, the Backup RTC is ideal for applications where keeping track of time in combination with extremely low energy consumption is essential. An overview of the backup RTC is shown in Figure 229.

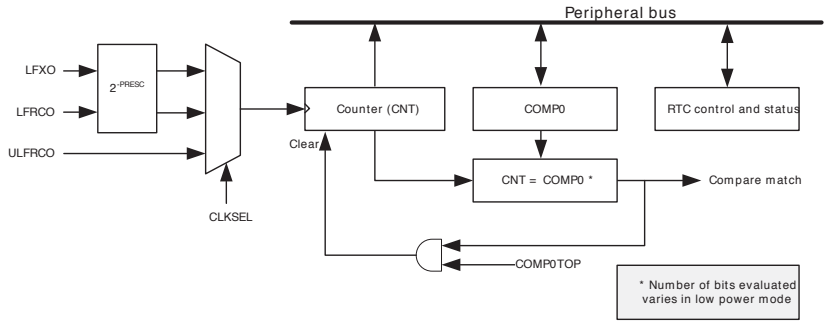


Figure 229: BURTC Overview

V.3.1 Counter

The Backup RTC is enabled by configuring MODE in the BURTC_CTRL register. This configuration of MODE determines in which energy modes the backup RTC is operational. It will always be operational in EM0-EM2, and optionally in EM3 and EM4. The Backup RTC is available when the system is in backup mode if MODE is set to EM4EN. The counter is cleared by setting RSTEN in the control register. A system reset will not clear the counter. The counter value can be read through the CNT register.

V.3.2 Clock source

The Backup RTC is clocked by LFXO, LFRCO, or ULFRCO, depending on the configuration of CLKSEL in BURTC_CTRL. The PRESC bit-field in BURTC_CTRL controls the clock prescaling factor. Prescaler is only available for LFXO and LFRCO. When using the ULFRCO as clock source, only two frequency options are available; 2kHz and 1kHz. The 2kHz clock is selected when PRESC in BURTC_CTRL is set to DIV1, and the 1kHz clock is selected when PRESC is set to any other value. Available frequencies when using LFXO or LFRCO are given in Equation V.3.2. CLKSEL should not be changed while the backup RTC is running.

$$f_{BURTC} = 32768/2^{PRESC}\text{Hz, PRESC} = 0..7\ (31)$$

When the LFXO or LFRCO is enabled, the Backup RTC will not use the clock until the timeout defined in the CMU has run out, i.e. the LFXORDY/LFRCORDY flag in CMU_STATUS is set. When an oscillator first has been enabled and is used by the Backup RTC, the Backup RTC will keep the selected clock source enabled, independent of both energy mode and CMU settings.

V.3.3 Compare channel

The backup RTC has one compare channel. The compare value is set by writing to the COMP0 register. When the value of CNT equals the value of COMP0, the COMP0 interrupt flag is set. If COMP0TOP in CTRL is set, the counter will wrap around when reaching the value in the compare register, COMP. If COMP0TOP is cleared, the counter will continue counting, wrapping around when it overflows. On overflow, the OF interrupt flag is set.

V.3.4 PRS Sources

The compare channel of the Backup RTC can be used as PRS source. A pulse lasting one clock cycle will be generated on a compare match. A PRS pulse will also be generated on overflow.

V.3.5 Debugrun

By default, the backup RTC is halted when code execution is halted by the debugger. By setting the DEBUGRUN bit in the CTRL register, the backup RTC will continue to run even when the system is halted.

V.3.6 Low power mode

The Backup RTC has a low power mode which lowers the power consumption at the expense of decreased resolution on compare matches. The low power mode is enabled by configuring the LPMODE bit-field in BURTC_CTRL. When LPMODE is set to ENABLE, low power mode is always enabled, if LPMODE is set to BUEN, the Backup RTC operates in normal mode until the system enters backup mode, refer to [?] for details on backup mode. When the Backup RTC operates in low power mode, a configurable number of the LSBs of COMP0 are ignored for compare match evaluation. The number of bits ignored is configured in the LPCOMP bit-field in the BURTC_CTRL register. Equation V.3.6 is used to calculate compare match resolution in low power mode.

In low power mode, the Backup RTC will decrease its frequency by a factor of $2^{-LPCOMP}$, and start incrementing with 2^{LPCOMP} instead of 1. When reading the counter value from software, full resolution is maintained, the decrease in frequency will only affect the resolution on compare matches. Low power mode can be entered and exited while the Backup RTC is running. When the Backup RTC is operating in low power mode, LPMODEACT in BURTC_STATUS is set.

$$CM_{resolution} = 2^{PRESC + LPCOMP + 1} / F_{CLK}, PRESC + LPCOMP + 1 < 9 \quad (32)$$

[2^PRESC] [2^-m]	Normal mode		Low power mode	
	Compare match resolution	Overflow	Compare match resolution	Overflow
0	30.5 μs	1.52 days	Equation V.3.6	1.52 days
1	61 μs	3.03 days	Equation V.3.6	3.03 days
2	122 μs	6.07 days	Equation V.3.6	6.07 days
3	244 μs	12.14 days	Equation V.3.6	12.14 days
4	488 μs	24.27 days	Equation V.3.6	24.27 days
5	977 μs	48.54 days	Equation V.3.6	48.54 days
6	1.95 ms	97.09 days	Equation V.3.6	97.09 days
7	3.91 ms	194.18 days	Equation V.3.6	194.18 days

Figure 230: Resolution and overflow

Low power mode is only available when using LFXO or LFRCO.

V.3.7 Retention Registers

The Backup RTC includes 128 x 32 bit registers with possible retention in all energy modes. The registers are accessible through the RETx_REG registers. Retention is by default enabled in EM0 through EM4. The registers can be shut off to save power by setting RAM in BURTC_POWERDOWN. Note that the retention registers cannot be accessed when RSTEN in BURTC_CTRL is set.

The retention registers are mapped to a RAM instance and have undefined state out of reset.

If the system should lose main power and enter backup mode while writing to the retention registers, the RAM write error flag, RAMWERR, in BURTC_STATUS will be set, and the attempted write will be canceled. The RAMWERR flag is cleared by writing a 1 to CLRSTATUS in BURTC_CMD.

V.3.8 Backup operation

The Backup RTC and the retention registers reside in a separate power domain, which in addition to being available in EM4 has the possibility to be powered by a backup battery. Refer to [?] for further details on this power domain.

V.3.9 Backup mode timestamp

The Backup RTC includes functionality for storing a timestamp when the system enters backup mode. The timestamp is stored in the BURTC_TIMESTAMP register and is stored two cycles after entering backup mode. If Low Power mode is enabled, ignored bits will not be stored in the timestamp register. Timestamping is enabled by setting BUMODETSEN in BURTC_CTRL. When a timestamp is stored, the BUMODETS bit in BUCTRL_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, by writing a 1 to CLRSTATUS in BURTC_CMD. Note that upon clearing this bit, the data in BURTC_TIMESTAMP is no longer valid.

V.3.10 LFXO failure detection

To be able to detect LFXO failure, the Backup RTC includes a five bit down counter with configurable top value. The top value is configured in TOP in BURTC_LFXOFDET. The counter starts at the top value and counts downwards on either LFRCO or ULFRCO, depending on the configuration of OSC in BURTC_LFXOFDET. When LFRCO is selected as clock for the down counter, it will be prescaled with a factor of $2^{PRESC + LPCOMP}$. The counter wraps to TOP when it reaches zero. If no LFXO clock has arrived since the last time the counter reached zero , the BURTC clock is changed to the clock source configured in OSC and the LFXOFAIL interrupt flag is set. Note that due to synchronization, the LFXO clock needs to arrive at least two cycles before the counter reaches zero.

V.3.11 Register access

Most Backup RTC configuration should not be changed while the counter is running, i.e. they should only be changed while RSTEN in BURTC_CTRL is set.

Registers allowed to change run-time are BURTC_COMP0, BURTC_LPMODE, and DEBUGRUN in BURTC_CTRL. For further details on access to these registers, refer to [?].

The Backup domain has its own reset signal which is active when the device powers up for the first time. The reset is deactivated by clearing BURSTEN in RMU_CTRL. This has to be done before any registers in the Backup RTC can be accessed.

V.4 Register Map

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

Offset	Name	Type	Description
0x000	BURTC_CTRL	RW	Control Register
0x004	BURTC_LPMODE	RW	Low power mode configuration
0x008	BURTC_CNT	R	Counter Value Register
0x00C	BURTC_COMP0	RW	Counter Compare Value
0x010	BURTC_TIMESTAMP	R	Backup mode timestamp
0x014	BURTC_LFXOFDET	RW	LFXO
0x018	BURTC_STATUS	R	Status Register
0x01C	BURTC_CMD	W1	Command Register
0x020	BURTC_POWERDOWN	RW	Retention RAM power-down Register
0x024	BURTC_LOCK	RW	Configuration Lock Register
0x028	BURTC_IF	R	Interrupt Flag Register
0x02C	BURTC_IFS	W1	Interrupt Flag Set Register
0x030	BURTC_IFC	W1	Interrupt Flag Clear Register
0x034	BURTC_IEN	RW	Interrupt Enable Register
0x038	BURTC_FREEZE	RW	Freeze Register
0x03C	BURTC_SYNCBUSY	R	Synchronization Busy Register
0x100	RET0_REG	RW	Retention Register
...	RETx_REG	RW	Retention Register
0x2FC	RET127_REG	RW	Retention Register

V.5 Register Description

V.5.1 BURTC_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			0	1	0	0x0	
Access																		RW	RW			RW			RW			RW	RW	RW	RW	
Name																		BUMODETSEN	CLKSEL			PRESC			LPCOMP			COMPOTOP	RSTEN	DEBUGRUN	MODE	

Bit	Name	Reset	Access	Description
31:15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14	BUMODETSEN	0	RW	**Backup mode timestamp enable**
	When set, the BURTC will store its counter value in the BURTC_TIMESTAMP register upon backup mode entry.			
13:12	CLKSEL	0x0	RW	**Select BURTC clock source**

Value	Mode	Description
0	NONE	No clock source selected for BURTC.
1	LFRCO	LFRCO selected as BURTC clock source.
2	LFXO	LFXO selected as BURTC clock source.
3	ULFRCO	ULFRCO selected as BURTC clock source.

11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	PRESC	0x0	RW	**Select BURTC prescaler factor**
	The BURTC will be prescaled by a factor of 2^{PRESC}			

Value	Mode	Description
0	DIV1	No prescaling.
1	DIV2	Prescaling factor of 2
2	DIV4	Prescaling factor of 4
3	DIV8	Prescaling factor of 8
4	DIV16	Prescaling factor of 16
5	DIV32	Prescaling factor of 32
6	DIV64	Prescaling factor of 64
7	DIV128	Prescaling factor of 128

7:5	LPCOMP	0x0	RW	**Low power mode compare configuration**
	This bit-field configures which bits to be evaluated for compare match in low power mode.			

Value	Mode	Description
0	IGN0LSB	Do not ignore any bits for compare match evaluation.
1	IGN1LSB	The LSB of the counter is ignored for compare match evaluation.
2	IGN2LSB	The two LSBs of the counter are ignored for compare match evaluation.
3	IGN3LSB	The three LSBs of the counter are ignored for compare match evaluation.
4	IGN4LSB	The four LSBs of the counter are ignored for compare match evaluation.
5	IGN5LSB	The five LSBs of the counter are ignored for compare match evaluation.
6	IGN6LSB	The six LSBs of the counter are ignored for compare match evaluation.
7	IGN7LSB	The seven LSBs of the counter are ignored for compare match evaluation.

4	COMP0TOP	0	RW	**Compare clear enable**
	When set, the counter wraps around when CNT equals COMP0			
3	RSTEN	1	RW	**Enable BURTC reset**
	Reset the Backup RTC. Register values are not reset.			
2	DEBUGRUN	0	RW	**Debug Mode Run Enable**
	Set this bit to keep the BURTC running during a debug halt.			

Value	Description
0	RTC is frozen in debug mode
1	RTC is running in debug mode

1:0	MODE	0x0	RW	**BURTC Enable**
	Configure in which energy modes the BURTC should keep running.			

Value	Mode	Description
0	DISABLE	The BURTC is disabled.
1	EM2EN	The BURTC is in normal operating mode, operating in EM0-EM2. Oscillators must be enabled in CMU for use.
2	EM3EN	The BURTC is enabled in EM0-EM3. Will prevent CMU from disabling used oscillators all the way down to EM3.
3	EM4EN	The BURTC is enabled in EM0-EM4. Will prevent CMU from disabling used oscillators all the way down to EM4.

V.5.2 BURTC_LPMODE - Low power mode configuration (Async Reg)

For more information about Asynchronous Registers please see F.4.

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

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1:0	LPMODE	0x0	RW	**Low power mode configuration.**

Value	Mode	Description
0	DISABLE	Low power mode is disabled.
1	ENABLE	Low power mode always enabled.
2	BUEN	Low power mode enabled in backup mode.

V.5.3 BURTC_CNT - Counter Value 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	R																															
Name	CNT																															

Bit	Name	Reset	Access	Description
31:0	CNT	0x00000000	R	**Counter Value**
	Gives access to the BURTC counter value.			

V.5.4 BURTC_COMP0 - Counter Compare Value (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	0x00000000																															
Access	RW																															
Name	COMP0																															

Bit	Name	Reset	Access	Description
31:0	COMP0	0x00000000	RW	**Compare match value**
	Gives access to the BURTC compare value.			

V.5.5 BURTC_TIMESTAMP - Backup mode timestamp

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	TIMESTAMP																															

Bit	Name	Reset	Access	Description
31:0	TIMESTAMP	0x00000000	R	**Backup mode timestamp.**
	Contains the timestamp stored upon backup mode entry.			

V.5.6 BURTC_LFXOFDET - LFXO

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							0x0	
Access																								RW							RW	
Name																								TOP							OSC	

Bit	Name	Reset	Access	Description
31:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8:4	TOP	0x00	RW	**LFXO failure counter top value.**
	LFXO failure counter will wrap to this value when reaching zero.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1:0	OSC	0x0	RW	**LFXO failure detection configuration.**
	Select oscillator for LFXO failure detection.			

Value	Mode	Description
0	DISABLE	LFXO failure detection disabled.
1	LFRCO	LFRCO used for LFXO failure detection.
2	ULFRCO	ULFRCO used for LFXO failure detection.

V.5.7 BURTC_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	0
Access																														R	R	R
Name																														RAMWERR	BUMODETS	LPMODEACT

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	RAMWERR	0	R	**RAM write error.**
	Set if backup mode is entered during a write to the retention RAM.			

Bit	Name	Reset	Access	Description
1	BUMODETS	0	R	**Timestamp for backup mode entry stored.**
	Set when a timestamp has been stored in BURTC_TIMESTAMP.			
0	LPMODEACT	0	R	**Low power mode active**
	Set when the BURTC is in low power mode			

V.5.8 BURTC_CMD - Command 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																																W1
Name																																CLRSTATUS

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	CLRSTATUS	0	W1	**Clear BURTC_STATUS register.**
	Clear RAMWERR and BUMODETS in BURTC_STATUS.			

V.5.9 BURTC_POWERDOWN - Retention RAM power-down 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																																RW
Name																																RAM

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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			

V.5.10 BURTC_LOCK - Configuration Lock Register

Offset	Bit Position																													
0x024	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0000															
Access																	RW															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	LOCKKEY	0x0000	RW	**Configuration Lock Key**
	Write any other value than the unlock code to lock BURTC_POWERDOWN, BURTC_CTRL, BURTC_LFXOFDET, and BURTC_IEN registers from editing. Write the unlock code to unlock. When reading the register, bit 0 is set when the lock is enabled.			

Bit	Name	Reset	Access	Description
	Mode	**Value**		**Description**
	Read Operation			
	UNLOCKED	0		BURTC_POWERDOWN, BURTC_CTRL, BURTC_LFXOFDET, and BURTC_IEN registers are unlocked
	LOCKED	1		BURTC_POWERDOWN, BURTC_CTRL, BURTC_LFXOFDET, and BURTC_IEN registers are locked
	Write Operation			
	LOCK	0		Lock BURTC_POWERDOWN, BURTC_CTRL, BURTC_LFXOFDET, and BURTC_IEN registers
	UNLOCK	0xAEE8		Unlock BURTC registers

V.5.11 BURTC_IF - Interrupt Flag 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
Access																														R	R	R
Name																														LFXOFAIL	COMP0	OF

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	LFXOFAIL	0	R	**LFXO failure Interrupt Flag**
	Set on LFXO failure.			
1	COMP0	0	R	**Compare match Interrupt Flag**
	Set on BURTC compare match.			
0	OF	0	R	**Overflow Interrupt Flag**
	Set on BURTC overflow.			

V.5.12 BURTC_IFS - Interrupt Flag Set 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
Access																														W1	W1	W1
Name																														LFXOFAIL	COMP0	OF

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	LFXOFAIL	0	W1	**Set LFXO fail Interrupt Flag**
	Write to 1 to set the LFXOFAIL interrupt flag			
1	COMP0	0	W1	**Set compare match Interrupt Flag**
	Write to 1 to set the COMP0 interrupt flag			
0	OF	0	W1	**Set Overflow Interrupt Flag**
	Write to 1 to set the OF interrupt flag			

V.5.13 BURTC_IFC - Interrupt Flag Clear 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
Access																														W1	W1	W1
Name																														LFXOFAIL	COMP0	OF

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	LFXOFAIL	0	W1	**Clear LFXO failure Interrupt Flag**
	Write to 1 to clear the LFXOFAIL interrupt flag			
1	COMP0	0	W1	**Clear compare match Interrupt Flag**
	Write to 1 to clear the COMP0 interrupt flag			
0	OF	0	W1	**Clear Overflow Interrupt Flag**
	Write to 1 to clear the OF interrupt flag			

V.5.14 BURTC_IEN - Interrupt 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
Access																														RW	RW	RW
Name																														LFXOFAIL	COMP0	OF

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	LFXOFAIL	0	RW	**LFXO failure Interrupt Enable**
	Enable interrupt on LFXO failure			
1	COMP0	0	RW	**Compare match Interrupt Enable**
	Enable interrupt on compare match			
0	OF	0	RW	**Overflow Interrupt Enable**
	Enable interrupt on overflow			

V.5.15 BURTC_FREEZE - Freeze 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
Access																																RW
Name																																REGFREEZE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		

Bit	Name	Reset	Access	Description
0	REGFREEZE	0	RW	**Register Update Freeze**

When set, the update of the BURTC 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 BURTC register is updated into the Low Frequency domain as soon as possible.
1	FREEZE	The BURTC is not updated with the new written value until the freeze bit is cleared.

V.5.16 BURTC_SYNCBUSY - Synchronization Busy 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
Access																															R	R
Name																															COMP0	LPMODE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	COMP0	0	R	**COMP0 Register Busy**
	Set when the value written to COMP0 is being synchronized.			
0	LPMODE	0	R	**LPMODE Register Busy**
	Set when the value written to LPMODE is being synchronized.			

V.5.17 RETx_REG - Retention 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	0xXXXXXXXX																													
Access	RW																													
Name	REG																													

Bit	Name	Reset	Access	Description
31:0	REG	0xXXXXXXXX	RW	**General Purpose Retention Register**

W ARM Low Energy Timer

W.1 Introduction

The unique LETIMER™, the Low Energy Timer, is a 16-bit timer that is available in energy mode EM2 and EM3, in addition to EM1 and EM0. 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 is also connected to the Real Time Counter (RTC), and can be configured to start counting on compare matches from the RTC.

W.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 or by software
- 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

W.3 Functional Description

An overview of the LETIMER module is shown in Figure 231. 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.

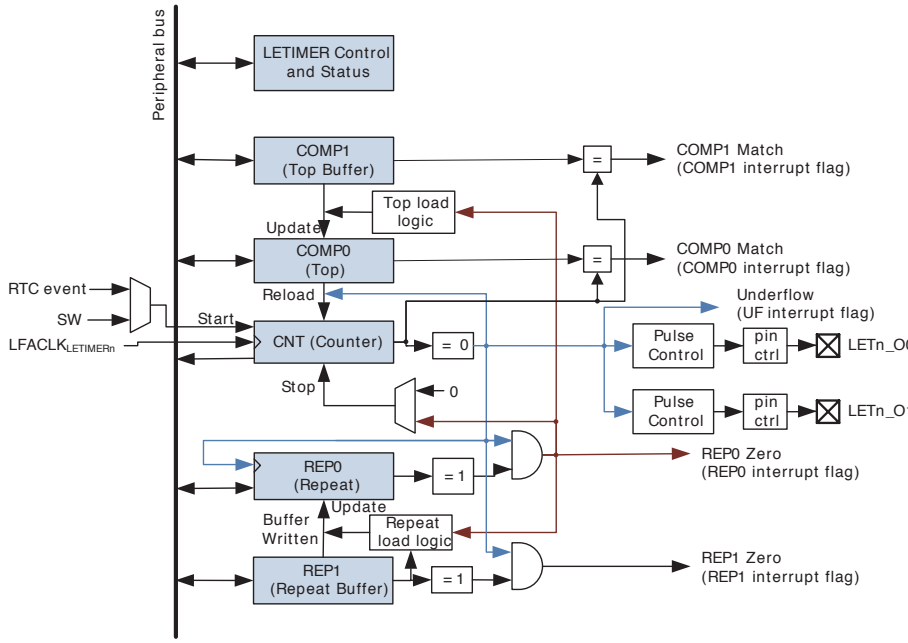


Figure 231: LETIMER Overview

W.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 cannot be written, but it can 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.

W.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.

W.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. Else, the timer wraps around to 0xFFFF. The underflow interrupt flag UF in LETIMERn_IF is set when the timer reaches zero.

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.

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 232.

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.

Free Mode In the 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 233 .

REPMODE	Mode	Description
00	Free	The timer runs until it is stopped
01	One-shot	The timer runs as long as LETIMERn_REP0 != 0. LETIMERn_REP0 is decremented at each timer underflow.
10	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.
11	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.

Figure 232: LETIMER Repeat Modes

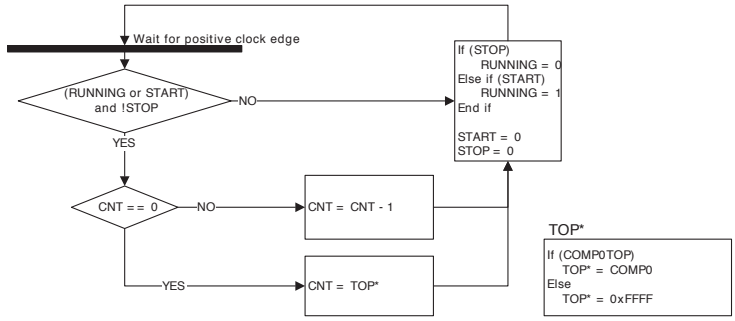


Figure 233: LETIMER State Machine for Free-running Mode

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.

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.

Write operations to LETIMERn_REP0 have priority over the decrementation operation. So if LETIMERn_REP0 is assigned a new value in the same cycle it was supposed to be decremented, it is assigned the new value instead of being decremented.

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 234 .

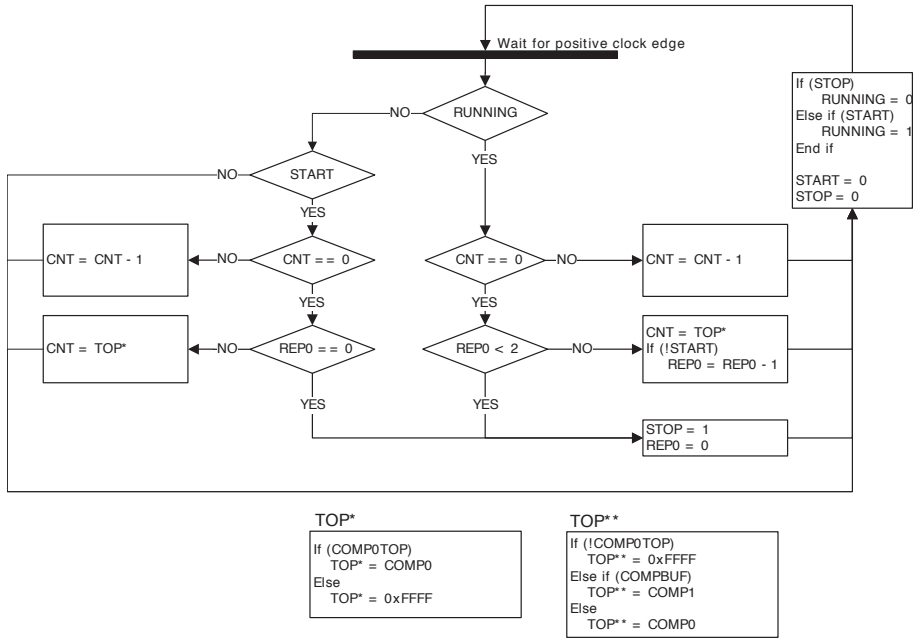


Figure 234: LETIMER One-shot Repeat State Machine

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.

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. The state machine for the one-shot repeat mode is shown in Figure 234.

Used in conjunction with a buffered top value, enabled by setting BUFTOP in LETIMERn_CTRL, the buffered mode allows buffered values of both the top and repeat values of the timer, 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 235. $REP1_{USED}$ 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.

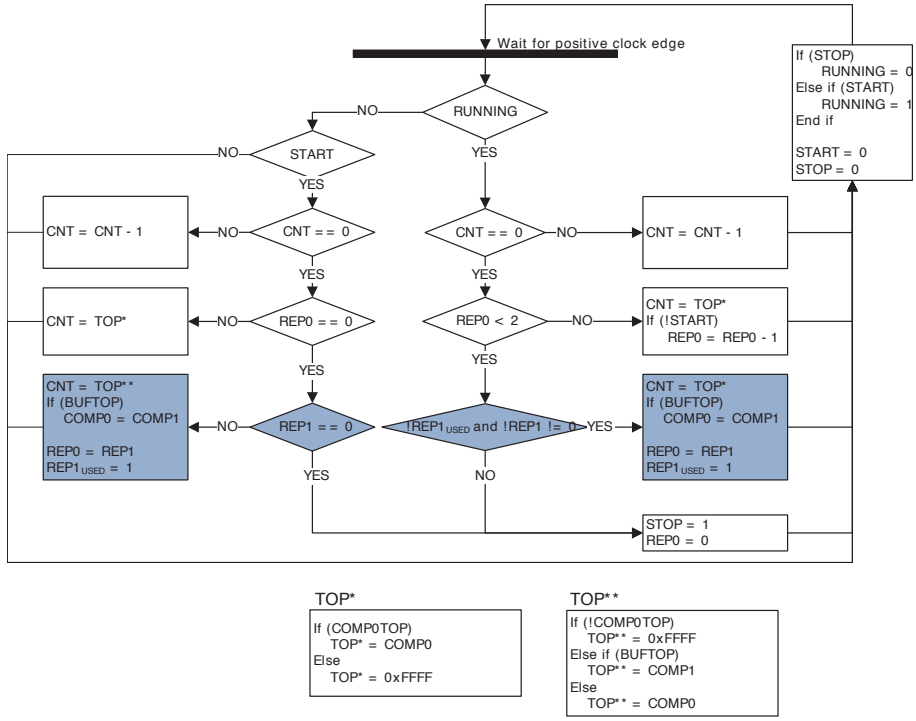


Figure 235: LETIMER Buffered Repeat State Machine

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 236.

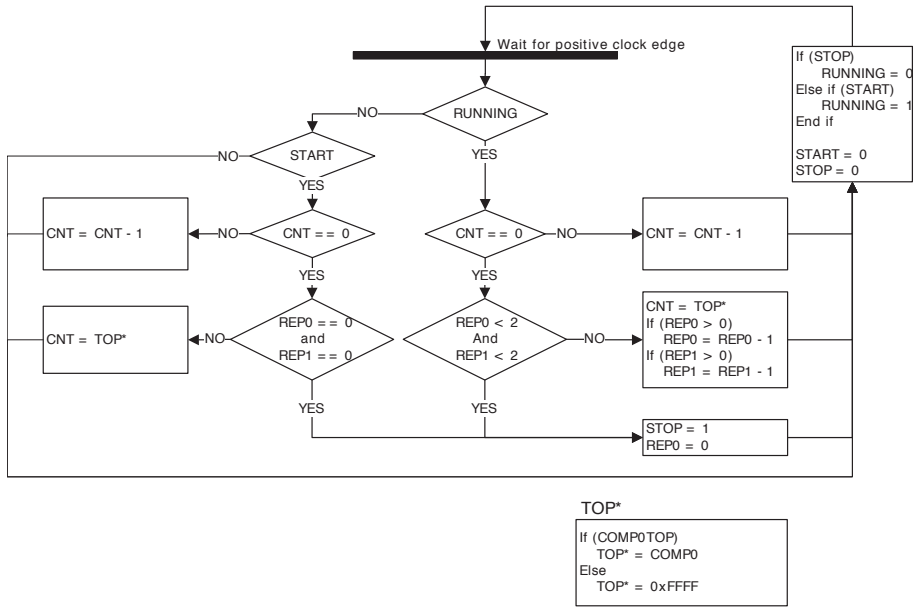


Figure 236: LETIMER Double Repeat State Machine

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 Equation 33.

$$f_{LFACKL\_LETIMERn} = 32.768/2^{LETIMERn} \quad (33)$$

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_HFCORECLKEN0, in addition to the module clock.

RTC Trigger

The LETIMER can be configured to start on compare match events from the Real Time Counter (RTC). If RTCC0TEN in LETIMERn_CTRL is set, the LETIMER will start on a compare match on RTC compare channel 0. In the same way, RTCC1TEN in LETIMERn_CTRL enables the LETIMER to start on a compare match with RTC compare channel 1.

The LETIMER can only use compare match events from the RTC if the LETIMER runs at a higher than or equal frequency than the RTC. Also, if the LETIMER runs

at twice the frequency of the RTC, a compare match event in the RTC will trigger the LETIMER twice. Four times the frequency gives four consecutive triggers, etc. The LETIMER will only continue running if triggered while it is running, so the multiple-triggering will only have an effect if you try to disable the RTC when it is being triggered.

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.

W.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 for a while.

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 237.

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.

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 238. 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

UF0A0/UF0A1	Mode	Description
00	Idle	The output is held at its idle value
01	Toggle	The output is toggled on LETIMERn_CNT underflow if LEIMERn_REPx is nonzero
10	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
11	PWM	The output is set idle on LETIMERn_CNT underflow and active on compare match with LETIMERn_COMP1 if LETIMERn_REPx is nonzero.

Figure 237: LETIMER Underflow Output Actions

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.

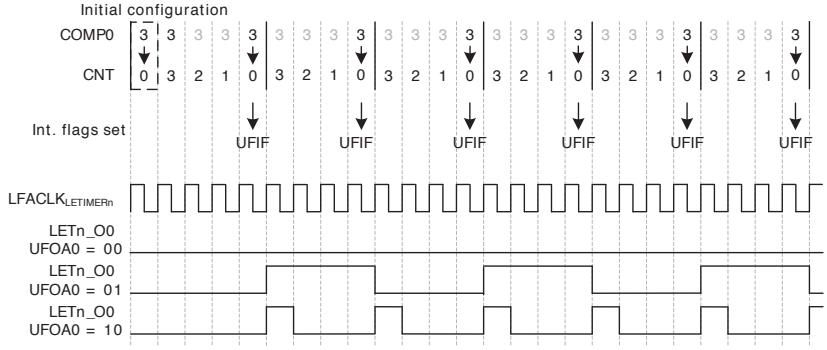


Figure 238: LETIMER Simple Waveforms Output

For the example in Figure 239, 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.

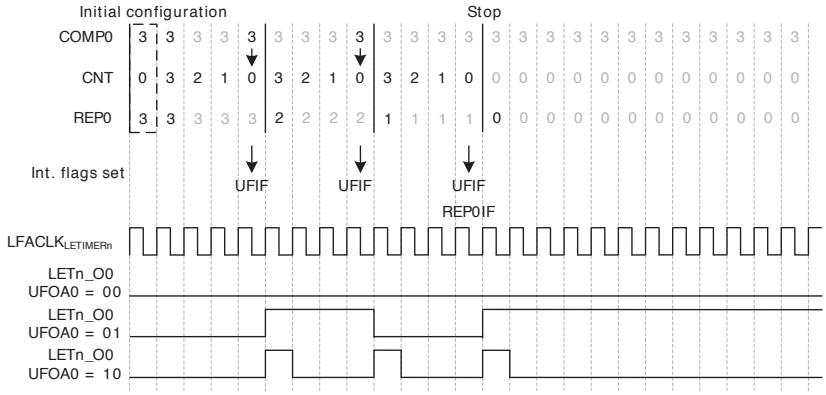


Figure 239: LETIMER Repeated Counting

Using the Double repeat mode, output can be generated on both the LETIMER outputs. Figure 240shows 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.

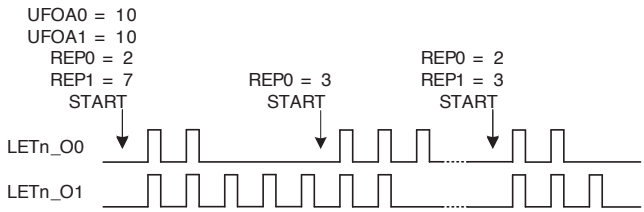


Figure 240: LETIMER Dual Output

W.3.5 PRS Output

The LETIMER outputs can be routed out onto the PRS system. LETn_O0 can be routed to PRS channel 0, and LETn_1O can be routed to PRS channel 1. Enabling the RRS 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 [?]

W.3.6 Examples

This section presents a couple of usage examples for the LETIMER.

Triggered Output Generation

exampleLETIMER 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 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 241, 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.

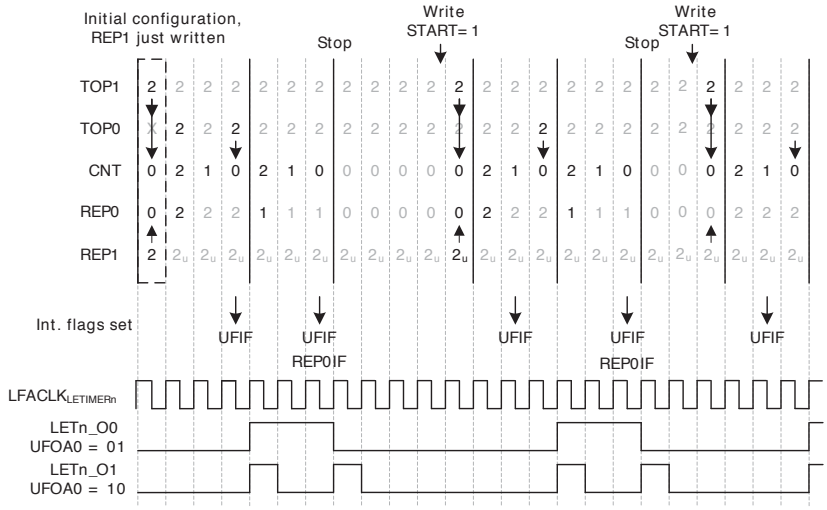


Figure 241: LETIMER Triggered Operation

Continuous Output Generation

exampleLETIMER 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 238, but to generate changing waveforms, using the repeat counter and buffer registers can prove advantageous.

For the example in Figure 242, 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

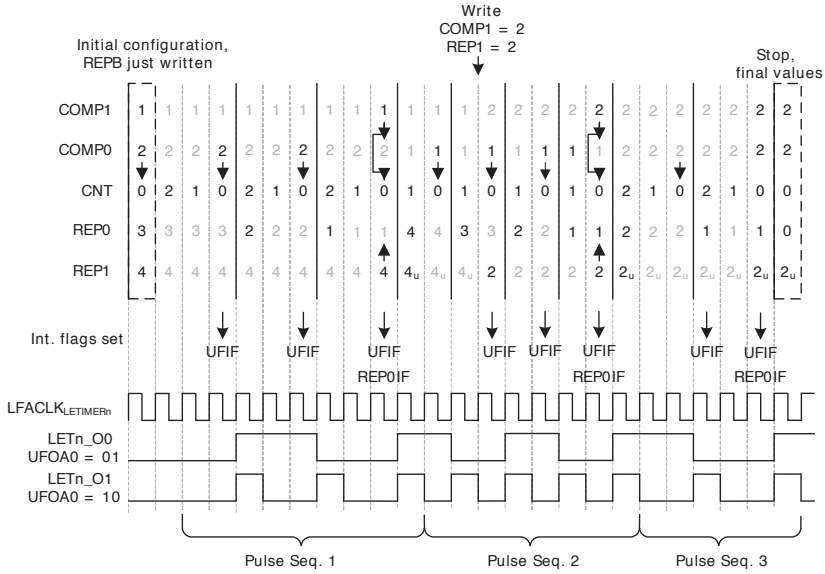


Figure 242: LETIMER Continuous Operation

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 242. 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.

Multiple LETIMER cycles are required to write a value to the LETIMER registers. The example in Figure 242 assumes that writes are done in advance so they arrive in the LETIMER as described in the figure.

Figure 243 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.

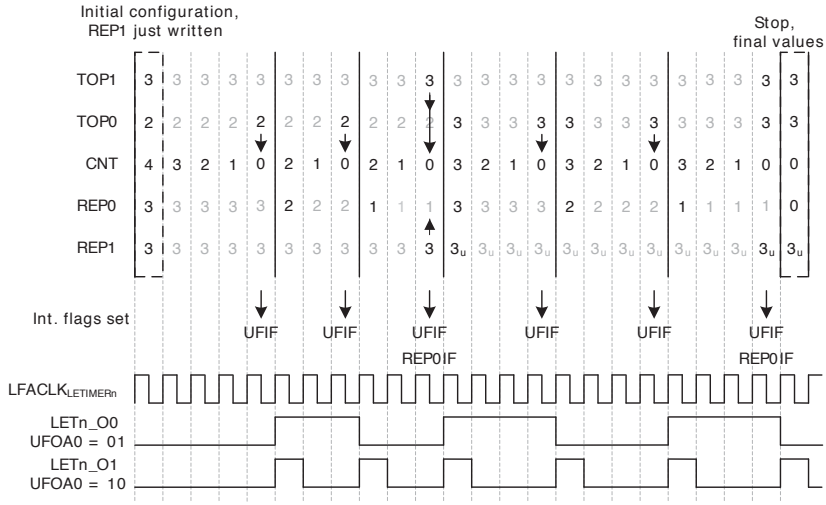


Figure 243: LETIMER LETIMERn_CNT Not Initialized to 0

PWM Output

exampleLETIMER 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 OFUA1 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_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.

Interrupts

exampleLETIMER PWM Output

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.

W.3.7 Using the LETIMER in EM3

The LETIMER can be enabled all the way down to EM3 by using the ULFRCO as clock source. This is done by clearing CMU_LFCLKSEL_LFA and setting CMU_LFCLKSEL_LFAE to 1. This will make the RTC use the internal 1 kHz ultra low frequency RC oscillator (ULFRCO), consuming very little energy. Please note that the ULFRCO is not accurate over temperature and voltage, and it should be verified that the ULFRCO fulfills the timekeeping needs of the application before using this in the design.

W.3.8 Register access

This module is a Low Energy Peripheral, and supports immediate synchronization.

W.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	RW	Compare Value Register 0
0x014	LETIMERn_COMP1	RW	Compare Value Register 1
0x018	LETIMERn_REP0	RW	Repeat Counter Register 0
0x01C	LETIMERn_REP1	RW	Repeat Counter Register 1
0x020	LETIMERn_IF	R	Interrupt Flag Register
0x024	LETIMERn_IFS	W1	Interrupt Flag Set Register
0x028	LETIMERn_IFC	W1	Interrupt Flag Clear Register
0x02C	LETIMERn_IEN	RW	Interrupt Enable Register
0x030	LETIMERn_FREEZE	RW	Freeze Register
0x034	LETIMERn_SYNCBUSY	R	Synchronization Busy Register
0x040	LETIMERn_ROUTE	RW	I/O Routing Register

W.5 Register Description

W.5.1 LETIMERn_CTRL - Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	0x0		0x0		0x0	
Access																				RW	RW	RW	RW	RW	RW	RW	RW		RW		RW	
Name																				DEBUGRUN	RTCC1TEN	RTCC0TEN	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.		
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	RTCC1TEN	0	RW	**RTC Compare 1 Trigger Enable**
	Allows the LETIMER to be started on a compare match on RTC compare channel 1.			

Value	Description
0	LETIMER is not affected by RTC compare channel 1
1	A compare match on RTC compare channel 1 starts the LETIMER if the LETIMER is not already started

Bit	Name	Reset	Access	Description
10	RTCC0TEN	0	RW	**RTC Compare 0 Trigger Enable**
	Allows the LETIMER to be started on a compare match on RTC compare channel 0.			

Value	Description
0	LETIMER is not affected by RTC compare channel 0
1	A compare match on RTC compare channel 0 starts the LETIMER if the LETIMER is not already started

Bit	Name	Reset	Access	Description
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.			
6	OPOL0	0	RW	**Output 0 Polarity**
	Defines the idle value of output 0.			
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. Else 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

W.5.2 LETIMERn_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
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.		
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			

W.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.		
0	RUNNING	0	R	**LETIMER Running**
	Set when LETIMER is running.			

W.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.		
15:0	CNT	0x0000	RWH	**Counter Value**
	Use to read the current value of the LETIMER.			

W.5.5 LETIMERn_COMP0 - Compare Value Register 0 (Async Reg)

For more information about Asynchronous Registers please see F.4.

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																	RW															
Name																	COMP0															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	COMP0	0x0000	RW	**Compare Value 0**
	Compare and optionally top value for LETIMER			

W.5.6 LETIMERn_COMP1 - Compare Value Register 1 (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
15:0	COMP1	0x0000	RW	**Compare Value 1**

Bit	Name	Reset	Access	Description
	Compare and optionally buffered top value for LETIMER			

W.5.7 LETIMERn_REP0 - Repeat Counter Register 0 (Async Reg)

For more information about Asynchronous Registers please see F.4.

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																									REP0							

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:0	REP0	0x00	RW	**Repeat Counter 0**
	Optional repeat counter.			

W.5.8 LETIMERn_REP1 - Repeat Counter Register 1 (Async Reg)

For more information about Asynchronous Registers please see F.4.

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																									RW							
Name																									REP1							

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:0	REP1	0x00	RW	**Repeat Counter 1**
	Optional repeat counter or buffer for REP0			

W.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.		
4	REP1	0	R	**Repeat Counter 1 Interrupt Flag**
	Set when repeat counter 1 reaches zero.			

Bit	Name	Reset	Access	Description
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			

W.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.		
4	REP1	0	W1	**Set Repeat Counter 1 Interrupt Flag**
	Write to 1 to set the REP1 interrupt flag.			
3	REP0	0	W1	**Set Repeat Counter 0 Interrupt Flag**
	Write to 1 to set the REP0 interrupt flag.			
2	UF	0	W1	**Set Underflow Interrupt Flag**
	Write to 1 to set the UF interrupt flag.			
1	COMP1	0	W1	**Set Compare Match 1 Interrupt Flag**
	Write to 1 to set the COMP1 interrupt flag.			
0	COMP0	0	W1	**Set Compare Match 0 Interrupt Flag**
	Write to 1 to set the COMP0 interrupt flag.			

W.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																												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.		
4	REP1	0	W1	**Clear Repeat Counter 1 Interrupt Flag**
	Write to 1 to clear the REP1 interrupt flag.			
3	REP0	0	W1	**Clear Repeat Counter 0 Interrupt Flag**
	Write to 1 to clear the REP0 interrupt flag.			
2	UF	0	W1	**Clear Underflow Interrupt Flag**
	Write to 1 to clear the UF interrupt flag.			

Bit	Name	Reset	Access	Description
1	COMP1	0	W1	**Clear Compare Match 1 Interrupt Flag**
	Write to 1 to clear the COMP1 interrupt flag.			
0	COMP0	0	W1	**Clear Compare Match 0 Interrupt Flag**
	Write to 1 to clear the COMP0 interrupt flag.			

W.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.		
4	REP1	0	RW	**Repeat Counter 1 Interrupt Enable**
	Set to enable interrupt on the REP1 interrupt flag.			
3	REP0	0	RW	**Repeat Counter 0 Interrupt Enable**
	Set to enable interrupt on the REP0 interrupt flag.			
2	UF	0	RW	**Underflow Interrupt Enable**
	Set to enable interrupt on the UF interrupt flag.			
1	COMP1	0	RW	**Compare Match 1 Interrupt Enable**
	Set to enable interrupt on the COMP1 interrupt flag.			
0	COMP0	0	RW	**Compare Match 0 Interrupt Enable**
	Set to enable interrupt on the COMP0 interrupt flag.			

W.5.13 LETIMERn_FREEZE - Freeze 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																																RW
Name																																REGFREEZE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	REGFREEZE	0	RW	**Register Update Freeze**
	With the immediate write synchronization scheme the REGFREEZE register is no longer used.			

Value	Mode	Description
0	UPDATE	Each write access to a LETIMER register is updated into the Low Frequency domain as soon as possible.
1	FREEZE	The LETIMER is not updated with the new written value.

W.5.14 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	0	0	0	0	0
Access																											R	R	R	R	R	R
Name																											REP1	REP0	COMP1	COMP0	CMD	CTRL

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5	REP1	0	R	**REP1 Register Busy**
	Set when the value written to REP1 is being synchronized.			
4	REP0	0	R	**REP0 Register Busy**
	Set when the value written to REP0 is being synchronized.			
3	COMP1	0	R	**COMP1 Register Busy**
	Set when the value written to COMP1 is being synchronized.			
2	COMP0	0	R	**COMP0 Register Busy**
	Set when the value written to COMP0 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.			

W.5.15 LETIMERn_ROUTE - I/O Routing Register

Offset	Bit Position																													
0x040	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																						0x0									0	0
Access																						RW									RW	RW
Name																						LOCATION									OUT1PEN	OUT0PEN

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the LETIMER I/O 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
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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**

Bit	Name	Reset	Access	Description
	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

X ARM Pulse Counter

X.1 Introduction

The Pulse Counter (PCNT) can be used for counting incoming pulses on a single input or to decode quadrature encoded inputs. It can run from the internal LFACLK (EM0-EM2) while counting pulses on the PCNTn_S0IN pin or using this pin as an external clock source (EM0-EM3) that runs both the PCNT counter and register access.

X.2 Features

- 16/8-bit counter with reload register
- Auxiliary counter for counting a single direction
- Single input oversampling up/down counter mode (EM0-EM2)
- Externally clocked single input pulse up/down counter mode (EM0-EM3)
- Externally clocked quadrature decoder mode (EM0-EM3)
- 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
- PRS S0IN and S1IN input

X.3 Functional Description

An overview of the PCNT module is shown in Figure 244.

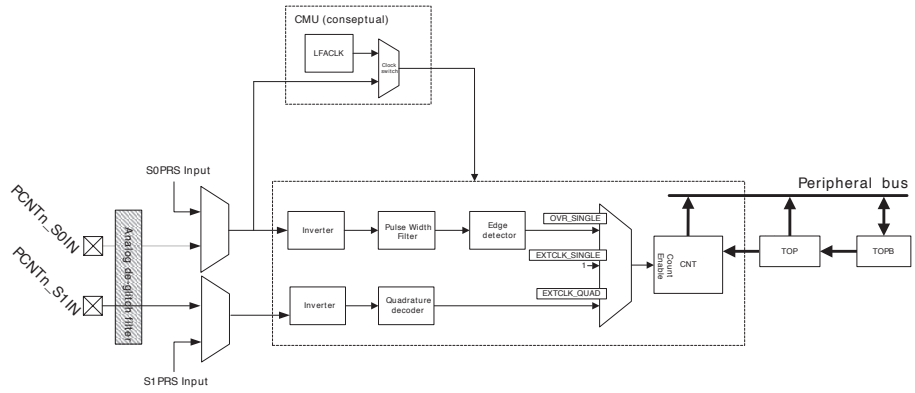


Figure 244: PCNT Overview

X.3.1 Pulse Counter Modes

The pulse counter can operate in single input oversampling mode (OVSSINGLE), externally clocked single input counter mode (EXTCLKSINGLE) and externally clocked quadrature decoder mode (EXTCLKQUAD). The following sections describe operation of each of the three modes and how they are enabled. Input timing constraints are described in Section X.3.5 and Section X.3.6.

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. LFACLK is configured from the registers 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 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 in PCNTn_CTRL. 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.

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 must be configured from the registers in the CMU.

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 in PCNTn_CTRL is set. If not, CNTDIR in PCNTn_CTRL solely defines count direction. As the LFACLK is not used in this mode, the PCNT module can operate in EM3.

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.

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 must be configured from the registers in the CMU.

Both edges on PCNTn_S0IN pin are used to sample PCNTn_S1IN pin to decode the quadrature code. Consequently, this mode does not depend on the internal LFACLK and may be operated in EM3. A quadrature coded signal contains information about the relative speed and direction of a rotating shaft as illustrated by Figure 245, hence the direction of the counter register PCNTn_CNT is controlled automatically.

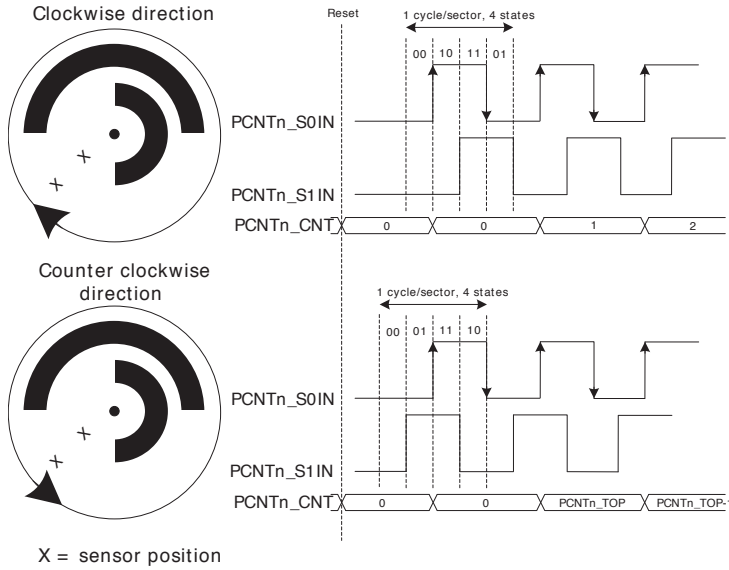


Figure 245: PCNT Quadrature Coding

If PCNTn_S0IN leads PCNTn_S1IN in phase, the direction is clockwise, and if it lags in phase the direction is counter-clockwise. Although the direction is automatically detected, the detected direction may be inverted by writing 1 to the EDGE bit in the PCNTn_CTRL register. Default behavior is illustrated by Figure 245.

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.

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 246. Note that this function also filters some invalid inputs that may occur when the shaft changes direction or temporarily toggles direction.

PCNTn_S1IN is sampled on both edges of PCNTn_S0IN.

Figure 246: PCNT QUAD Mode Counter Control Function

Inputs		Control/Status	
S1IN posedge	S1IN negedge	Count Enable	CNTDIR status bit
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0

X.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. If you have the latter however, and the counter changes directions around the overflow/underflow point, the system will have to wake up a lot to keep track of the rotations, causing high current consumptions

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.

Given a starting value of 0 for the counter, the absolute count value when hysteresis is enabled can be calculated with the equations Equation 34 or Equation 35, depending on whether the TOP value is even or odd.

$$CNT_{\text{abs}} = CNT - UF_{\text{CNT}} x (TOP/2 + 1) + OF_{\text{CNT}} x (TOP/2 + 1) \quad (34)$$

$$CNT_{\text{abs}} = CNT - UF_{\text{CNT}} x (TOP/2 + 1) + OF_{\text{CNT}} x (TOP/2 + 2) \quad (35)$$

X.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, and 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.

X.3.4 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.

When the RSTEN bit in the PCNTn_CTRL register is set to 1, 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.

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. Please refer to [?] for a description on how to perform register accesses to Low Energy Peripherals.

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.

X.3.5 Clock Sources

The 32 kHz LFACLK is one of two possible clock sources. The clock select register is described in [?]. 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 datasheet.

To use this module, the LE interface clock must be enabled in CMU_HFCORECLKEN0, in addition to the module clock.

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.

X.3.6 Input Filter

An optional pulse width filter is available in OVSSINGLE mode. 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 must be stable for 5 consecutive clock cycles before the edge is passed to the edge detector.

In EXTCLKSINGLE and EXTCLKQUAD mode, there is no digital pulse width filter available.

X.3.7 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 in OVSSINGLE mode and negative edges if the bit is set.

In EXTCLKQUAD mode, the EDGE bit in PCNTn_CTRL inverts the direction of the counter (which is automatically detected).

The EDGE bit in PCNTn_CTRL has no effect in EXTCLKSINGLE mode.

X.3.8 PRS S0IN and S1IN Input

It is possible to receive input from PRS on both S0IN and S1IN by setting S0PRSEN or S1PRSEN in PCNTn_INPUT. The PRS channel used can be selected using S0PRSSEL in PCNTn_INPUT.

X.3.9 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.

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.

Direction Change Interrupt

The PCNTn_PCNT module sets the DIRCNG interrupt flag (PCNTn_IF register) when the direction of the quadrature code changes. The behavior of this interrupt is illustrated by Figure 247.

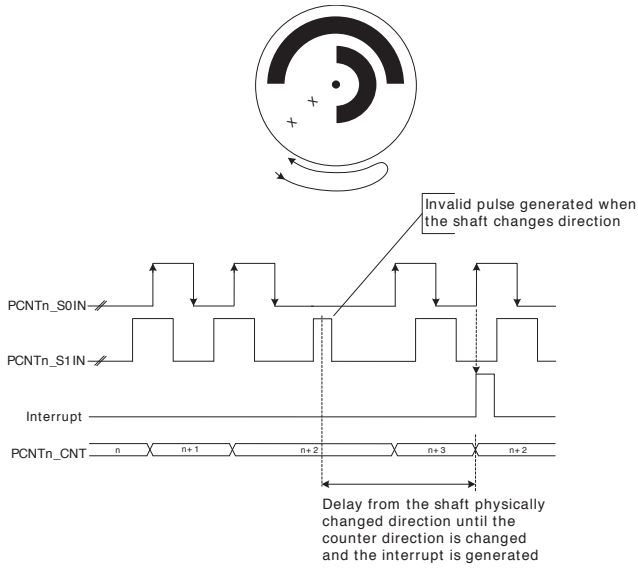


Figure 247: PCNT Direction Change Interrupt (DIRCNG) Generation

X.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	W1	Interrupt Flag Clear Register
0x024	PCNTn_IEN	RW	Interrupt Enable Register
0x028	PCNTn_ROUTE	RW	I/O Routing Register
0x02C	PCNTn_FREEZE	RW	Freeze Register
0x030	PCNTn_SYNCBUSY	R	Synchronization Busy Register
0x038	PCNTn_AUXCNT	RWH	Auxiliary Counter Value Register
0x03C	PCNTn_INPUT	RW	PCNT Input Register

X.5 Register Description

X.5.1 PCNTn_CTRL - Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	0	0	0x0	
Access																	RW				RW		RW	RW			RW	RW	RW	RW	RW	
Name																	AUXCNTEV				CNTEV		S1CDIR	HYST			RSTEN	FILT	EDGE	CNTDIR	MODE	

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:14	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.

13:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.

7:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5	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 external clock used the reset should be performed by setting and clearing the bit without pending for SYNCBUSY bit.

4	FILT	0	RW	**Enable Digital Pulse Width Filter**

The filter passes all high and low periods that are at least 5 clock cycles long. This filter is only available in OVSSINGLE mode.

3	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 is ignored in EXTCLKSINGLE mode.

Value	Mode	Description
0	POS	Positive edges on the PCNTn_S0IN inputs are counted in OVSSINGLE mode.
1	NEG	Negative edges on the PCNTn_S0IN inputs are counted in OVSSINGLE mode, and the counter direction is inverted in EXTCLKQUAD mode.

2	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.

1:0	MODE	0x0	RW	**Mode Select**

Bit	Name	Reset	Access	Description

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-EM2).
2	EXTCLKSINGLE	Externally clocked single input counter mode (available in EM0-EM3).
3	EXTCLKQUAD	Externally clocked quadrature decoder mode (available in EM0-EM3).

X.5.2 PCNTn_CMD - Command Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
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.			

X.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.		
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 NEDGE bit in PCNTn_CTRL set to 0).
1	DOWN	Down counter mode.

X.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

Offset	Bit Position	
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.		
15:0	CNT	0x0000	R	**Counter Value**
	Gives read access to the counter.			

X.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.		
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.			

X.5.6 PCNTn_TOPB - Top Value Buffer Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
15:0	TOPB	0x00FF	RW	**Counter Top Buffer**
	Loaded automatically to TOP when written.			

X.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
Access																													R	R	R	R

Offset	Bit Position																															
Name																													AUXOF	DIRCNG	OF	UF

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	AUXOF	0	R	**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			

X.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
Access																													W1	W1	W1	W1
Name																													AUXOF	DIRCNG	OF	UF

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	AUXOF	0	W1	**Auxiliary Overflow Interrupt Set**
	Write to 1 to set the auxiliary overflow interrupt flag			
2	DIRCNG	0	W1	**Direction Change Detect Interrupt Set**
	Write to 1 to set the direction change interrupt flag			
1	OF	0	W1	**Overflow Interrupt Set**
	Write to 1 to set the overflow interrupt flag			
0	UF	0	W1	**Underflow interrupt set**
	Write to 1 to set the underflow interrupt flag			

X.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
Access																													W1	W1	W1	W1
Name																													AUXOF	DIRCNG	OF	UF

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		

Bit	Name	Reset	Access	Description
3	AUXOF	0	W1	**Auxiliary Overflow Interrupt Clear**
	Write to 1 to clear the auxiliary overflow interrupt flag			
2	DIRCNG	0	W1	**Direction Change Detect Interrupt Clear**
	Write to 1 to clear the direction change detect interrupt flag			
1	OF	0	W1	**Overflow Interrupt Clear**
	Write to 1 to clear the overflow interrupt flag			
0	UF	0	W1	**Underflow Interrupt Clear**
	Write to 1 to clear the underflow interrupt flag			

X.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
Access																													RW	RW	RW	RW
Name																													AUXOF	DIRCNG	OF	UF

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3	AUXOF	0	RW	**Auxiliary Overflow Interrupt Enable**
	Enable the auxiliary overflow interrupt			
2	DIRCNG	0	RW	**Direction Change Detect Interrupt Enable**
	Enable the direction change detect interrupt.			
1	OF	0	RW	**Overflow Interrupt Enable**
	Enable the overflow interrupt			
0	UF	0	RW	**Underflow Interrupt Enable**
	Enable the underflow interrupt			

X.5.11 PCNTn_ROUTE - I/O Routing Register

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

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	LOCATION	0x0	RW	**I/O Location**
	Defines the location of the PCNT input pins. E.g. PCNTn_S0#0, #1 or #2.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2
3	LOC3	Location 3

Bit	Name	Reset	Access	Description
7:0	Reserved	To ensure compatibility with future devices, always write bits to 0.		

X.5.12 PCNTn_FREEZE - Freeze 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																																REGFREEZE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.

X.5.13 PCNTn_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	0	0
Access																														R	R	R
Name																														TOPB	CMD	CTRL

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

X.5.14 PCNTn_AUXCNT - Auxiliary Counter Value Register

Offset	Bit Position																													
0x038	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																	0x0000															
Access																	RWH															
Name																	AUXCNT															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	AUXCNT	0x0000	RWH	**Auxiliary Counter Value**
	Gives read access to the auxiliary counter.			

X.5.15 PCNTn_INPUT - PCNT Input 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	0x0					0	0x0			
Access																						RW	RW					RW	RW			
Name																						S1PRSEN	S1PRSSEL					S0PRSEN	S0PRSSEL			

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10	S1PRSEN	0	RW	**S1IN PRS Enable**
	When set, the PRS channel is selected as input to S1IN.			
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.

5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4	S0PRSEN	0	RW	**S0IN PRS Enable**
	When set, the PRS channel is selected as input to S0IN.			
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.

Y ARM Low Energy Sensor Interface

Y.1 Introduction

LESENSE is a low energy sensor interface which utilizes on-chip peripherals to perform measurement of a configurable set of sensors. The results from sensor measurements can be processed by the LESENSE decoder, which is a configurable state machine with up to 16 states. The results can also be stored in a result buffer to be collected by CPU or DMA for further processing.

LESENSE operates in EM2, in addition to EM1 and EM0, and can wake up the CPU on configurable events.

Y.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.
- Support for multiple sensor types
  - LC sensors
  - Capacitive sensing
  - General analog sensors

Y.3 Functional description

LESENSE is a module 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, for measurement of sensor signals. LESENSE can also control the DAC to generate accurate reference voltages. Figure 248 shows an overview of the LESENSE module. LESENSE consists of a sequencer, count and compare block, a decoder, and a RAM block used for configuration and result storage. The sequencer handles interaction with other peripherals as well as timing of sensor measurements. The count and compare block is used to count pulses from ACMP outputs before comparing with a configurable threshold. To autonomously analyze sensor results, the LESENSE decoder provides possibility to define a finite state machine with up to 16 states, and programmable actions upon state transitions. This allows the decoder to implement a wide range of decoding schemes, for instance 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.

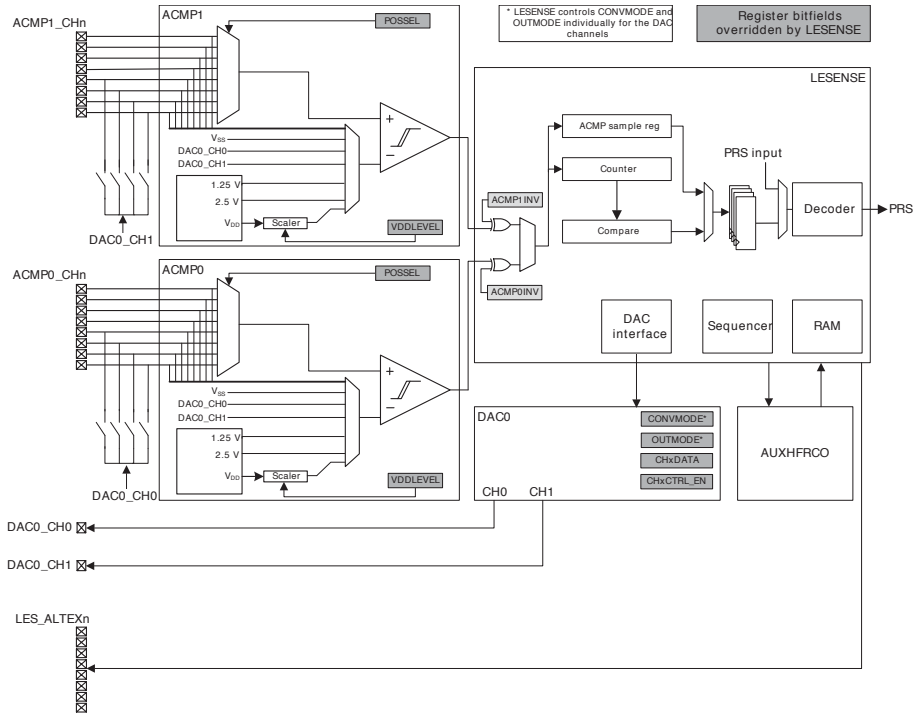


Figure 248: LESENSE block diagram

Y.3.1 Channel configuration

LESENSE has 16 individually configurable channels, the first eight are mapped to the channels of ACMP0, while the last eight are mapped to the channels of ACMP1. Each LESENSE channel has 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 address 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 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 for implementation of hysteresis on channel events as threshold values can be changed depending on 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 8 or higher, channel x will use CH_{X+8}_CONF, otherwise it will use CH_X configuration. Similarly, channels 8 through 15 will use CH_X configuration when decoder state index is less than 8 and CH_{X-8}_CONF when decoder state index is higher than 7. Allowing the decoder state to define which configuration to use, enables easy implementation of for instance hysteresis, as different threshold values can be used for the same channel, depending on the state of the application. Table 249 summarizes how channel configuration is selected for different setting of SCANCONF.

Figure 249: LESENSE scan configuration selection

LESENSE channel x	SCANCONF					
	DIRMAP	INVMAP	TOGGLE		DECDEF	
			SCANRES[n] = 0	SCANRES[n] = 1	DECSTATE < 8	DECSTATE >= 8
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 250 illustrates a scan sequence with channels 3, 5, and 9 enabled.

Y.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 Equation 36. 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 START in CMD, 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.

$$F_{\text{scan}} = LFACLK_{\text{LESENSE}} / ((1 + PCTOP) * 2^{\text{PCPRESC}}) \quad (36)$$

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.

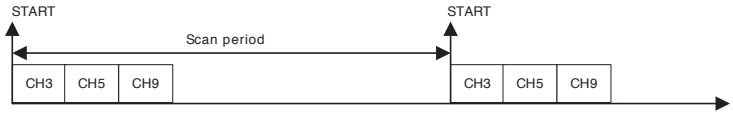


Figure 250: Scan sequence

Y.3.3 Sensor timing

For each channel in the scan sequence, the LESENSE interface goes through three phases: Idle phase, excite phase, and measure phase. The durations of the excite and measure phases are configured in the CHx_TIMING registers. LESENSE includes two timers: A low frequency timer, running on $LFACLK_{LESENSE}$, and a high frequency timer, running on AUXHFRCO. Timing of the excite phase is done using these timers and can be either a number of prescaled AUXHFRCO cycles or a number of prescaled $LFACLK_{LESENSE}$ cycles, depending on which one is selected in EXCLK. The prescaling can be done by configuring LFPRESC in TIMCTRL for the low frequency timer, and the high frequency timer prescaling factor is configured in AUXPRESC in the same register. The duration of the measure phase is programmed via MEASUREDLY and SAMPLEDLY. The output of the ACMP will be inactive 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 SAMPLECLK. Figure 251 depicts a sensor sequence where excitation and measure delay is timed using AUXHFRCO and the sample delay is timed using $LFACLK_{LESENSE}$. 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 DAC is done and voltages have stabilized before sensor measurement begins.

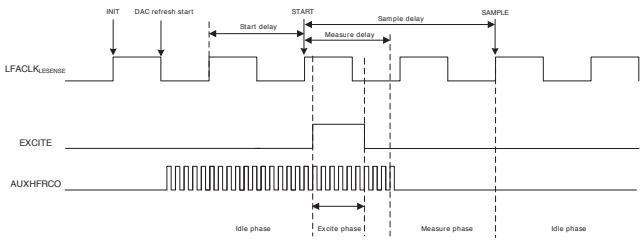


Figure 251: Timing diagram, short excitation

Y.3.4 Sensor interaction

Many sensor types require some type of excitation in order to work. LESENSE can generate a variety of sensor stimuli, both on the same pin as the measurement is to be made on, and on 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 in EXMODE 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 DAC channel.

Excitation with DAC output is only available on channels 0, 1, 2, and 3 (DAC0_CH0) and channels 12, 13, 14, and 15 (DAC0_CH1).

If the DAC 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 another pin than the one to be measured. When ALTEX in CHx_INTERACT is set, the excitation will occur on the alternative excite pin associated with the given channel. All LESENSE channels mapped to ACMP0 have their alternative channel mapped to the corresponding channel on ACMP1, and vice versa. Alternatively, the alternative excite pins can be routed to the LES_ALTEX pins. Mapping of the alternative excite pins is configured in ALTEXMAP in CTRL. Table 252 summarizes the mapping of excitation pins for different configurations.

Figure 253 illustrates the sequencing of the pin associated with the active channel and its alternative excite pin.

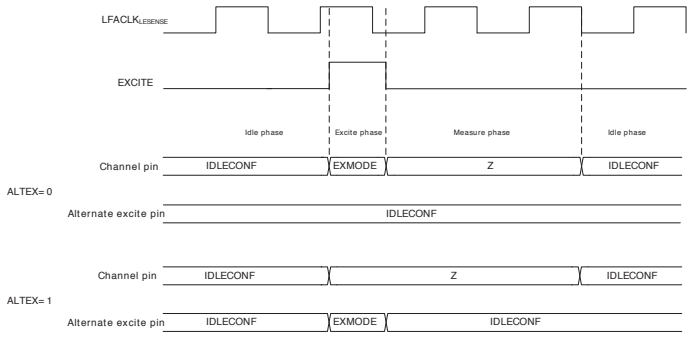


Figure 253: Pin sequencing

LESENSE channel	ALTEX = 0	ALTEX = 1	
		ALTEXMAP = ACMP	ALTEXMAP = ALTEX
0	ACMP0_CH0	ACMP1_CH0	LES_ALTEX0
1	ACMP0_CH1	ACMP1_CH1	LES_ALTEX1
2	ACMP0_CH2	ACMP1_CH2	LES_ALTEX2
3	ACMP0_CH3	ACMP1_CH3	LES_ALTEX3
4	ACMP0_CH4	ACMP1_CH4	LES_ALTEX4
5	ACMP0_CH5	ACMP1_CH5	LES_ALTEX5
6	ACMP0_CH6	ACMP1_CH6	LES_ALTEX6
7	ACMP0_CH7	ACMP1_CH7	LES_ALTEX7
8	ACMP1_CH0	ACMP0_CH0	LES_ALTEX0
9	ACMP1_CH1	ACMP0_CH1	LES_ALTEX1
10	ACMP1_CH2	ACMP0_CH2	LES_ALTEX2
11	ACMP1_CH3	ACMP0_CH3	LES_ALTEX3
12	ACMP1_CH4	ACMP0_CH4	LES_ALTEX4
13	ACMP1_CH5	ACMP0_CH5	LES_ALTEX5
14	ACMP1_CH6	ACMP0_CH6	LES_ALTEX6
15	ACMP1_CH7	ACMP0_CH7	LES_ALTEX7

Figure 252: LESENSE excitation pin mapping

The alternative excite pins, LES_ALTEXn, have the possibility 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.

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 enable LESENSE to control GPIO, the pin has to be enabled in the ROUTE register. In addition, the given pin must be configured as push-pull. IDLECONF configuration should not be altered when pin enable for the given pin is set in ROUTE.

Y.3.5 Sensor evaluation

Sensor evaluation can be based on either analog comparator outputs, or the counter output. This is configured in the SAMPLE bit-field in CHx_INTERACT. The LESENSE counter is used to count pulses on the ACMP output in the measurement phase. When a measurement phase is completed, the counter value is compared to the value configured in COMPTHRES in CHx_EVAL. By configuring COMP, it is possible to choose comparison mode: Less than, or greater than or equal. If a comparison for a channel triggers, the corresponding bit in the result register, SCANRES, is set. To set an interrupt flag on a sensor event, configure SETIF in CHx_INTERACT. Figure 254 illustrates how the counter value or ACMP sample is used for evaluation and interrupt generation.

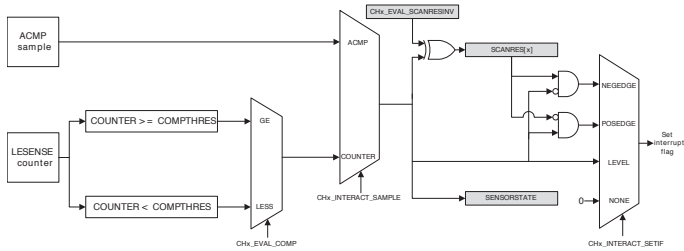


Figure 254: Scan result and interrupt generation

LESENSE includes the possibility 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, the channels of ACMP0 are paired together with the corresponding channel on ACMP1, i.e. channel x on ACMP0 and channel x on ACMP1 are sampled simultaneously. The results from sensor measurements can be fed into the decoder register and/or stored in the result buffer. In this mode, the samples from the AMCPs are placed in the two LSBs of the result stored in the result buffer. Results from both ACMPs will be evaluated for interrupt generation.

Y.3.6 Decoder

Many applications require some sort of processing of the sensor readings, for instance in the case of quadrature decoding. In quadrature decoding, the sensors repeatedly pass through a set of states which corresponds 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, LESENSE 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 16 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 255, where the bottom blocks show how the SENSORSTATE register is filled. When the scan sequence is complete, the decoder evaluates the state of the sensors chosen for decoding, as depicted in Figure 255.

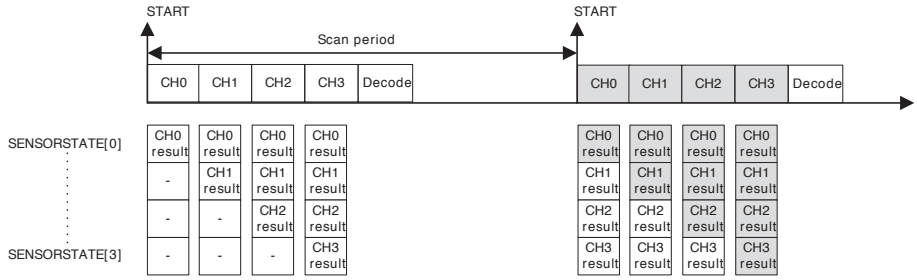


Figure 255: Sensor scan and decode sequence

The decoder is a programmable state machine with support for up to 16 states. The behavior of each state is individually configured in the STx_TCONFA and STx_TCONFB registers. The registers define possible transitions from the present state. If the sensor state matches COMP in either STx_TCONFA or STx_TCONFB, a transition to the state defined in NEXTSTATE will be made. It is also possible to mask out one or more sensors using the MASK bit field. The state of a masked sensor is interpreted as don't care.

Upon a state transition, LESENSE can generate a pulse on one or more of the decoder PRS channels. Which 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 ARM core 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 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 matches, 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 256 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.

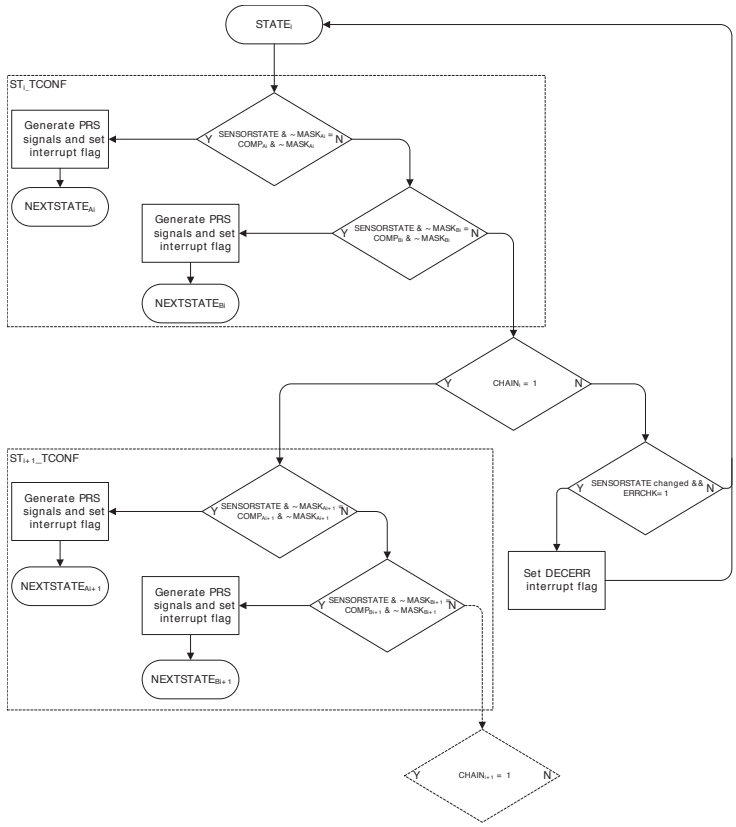


Figure 256: Decoder state transition evaluation

If only one transition from a state is used, STx_TCONFA and STx_TCONFB should be configured equally.

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 a transition defined in STx_TCONFA, and a type B transition is a transition defined in STx_TCONFB. When descriptor chaining is used, a jump to another descriptor will cancel out the hysteresis effect. Figure 257 illustrates how the hysteresis triggers upon state transitions.

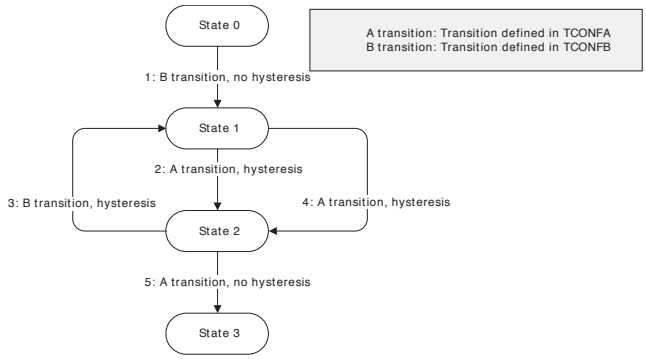


Figure 257: Decoder hysteresis

The events suppressed by the hysteresis are configured in bit fields HYSTPRS0-2 and HYSTIRQ in DECCTRL.

- 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.

The decoder error interrupt flag, DECERR, is not affected by the hysteresis.

Y.3.7 Measurement results

Part of the LESENSE RAM is treated as a circular buffer for storage of up to 16 results from sensor measurements. 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, is the oldest unread result. The location pointers are available in PTR. The result buffer has three status flags; 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 interrupt flag 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. If the result buffer overflows, the BUFOF interrupt flag will be set.

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. If SAMPLE is set to ACMP, the sensor is said to be triggered if the output from the analog comparator is 1 when sensor sampling is performed. If SAMPLE is set to COUNTER, a sensor is said to be triggered if the LESENSE counter value is greater than or equal, or less than COMPTHRES, depending on the configuration of COMP. If STRSAMPLE

in CHx_EVAL is set, the counter value or ACMP sample 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.

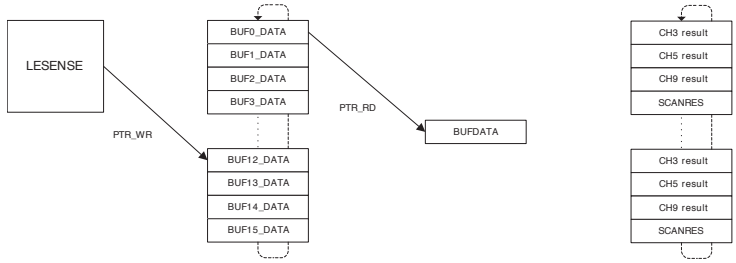


Figure 258: Circular result buffer

The right hand side of Figure 258 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.

Y.3.8 DAC interface

LESENSE is able to drive the DAC for generation of accurate reference voltages. DAC channels 0 and 1 are individually configured in the PERCTRL register. The conversion mode can be set to either continuous, sample/hold or sample/off. For further details about these modes, refer to [?]. Both DAC channels are refreshed prior to each sensor measurement, as depicted in Figure 251. The conversion data is either taken from the data registers in the DAC interface (DAC0_CH0DATA and DAC0_CH1DATA) or from the ACMPTHRES bit-field in the CHx_INTERACT register for the active LESENSE channel. DAC data used is configured in DACCHxDATA in PERCTRL.

The DAC interface runs on AUXHFRCO and will enable this when it is needed. The DACPRESC bit-field in PERCTRL is used to prescale the AUXHFRCO to achieve wanted clock frequency for the LESENSE DAC interface. The frequency should not exceed 500kHz, i.e. DACPRESC has to be set to at least 1. The prescaler may also be used to tune how long the DAC should drive its outputs in sample/off mode.

Bias configuration, calibration and reference selection is done in the DAC module and LESENSE will not override these configurations. If a bandgap reference is selected for the DAC, the DACREF bit in PERCTRL should be set to BANDGAP.

LESENSE has the possibility to control switches that connect the DAC outputs to the pins associated with ACMP0_CH0-3 and ACMP1_CH12-15. This makes LESENSE able to excite sensors with output from the DAC channels.

The DAC may be chosen as reference to the analog comparators for accurate reference generation. If the DAC is configured in continuous or sample/hold mode this does not require any external components. If sample/off mode is used, an external capacitor is needed to keep the voltage in between samples. To connect the input from the DAC to the ACMP to this external capacitor, connect the capacitor to the DAC pin for the given channel and set OPAxSHORT in DAC_OPACTRL.

The DAC mode should not be altered while DACACTIVE in STATUS is set

Y.3.9 ACMP interface

The 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 Vdd scaling factor (VDDLEVEL) for ACMP0 and ACMP1. The remaining configuration of the analog comparators are done in the ACMP register interface. It is recommended to set the MUXEN bit in ACMPn_CTRL for the ACMPs used by LESENSE. Each channel has the possibility to control the value of the Vdd scaling factor on the negative input of the ACMP, VDDLEVEL in ACMP_INPUTSEL. This is done in the 6 LSBs of ACMPTHRES in CHx_INTERACT. LESENSE automatically controls the ACMP mux to connect the correct channel.

Y.3.10 ACMP and DAC duty cycling

By default, the analog comparators and DAC are shut down in between LESENSE scans to save energy. If this is not wanted, WARMUPMODE in PERCTRL can be configured to prevent them from being shut down.

Both the DAC 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.

Y.3.11 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.

The DMA controller should always fetch data from the BUFDATA register.

Y.3.12 PRS output

LESENSE is an asynchronous PRS producer and has nineteen PRS outputs. The decoder has three 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 Section Y.3.6.

Y.3.13 RAM

LESENSE includes a RAM block used for storage of configuration and results. If LESENSE is not used, this RAM block can be powered down eliminating its current consumption due to leakage. The RAM is powered down by setting the RAM bit in the POWERDOWN register. Once the RAM has been shut down it cannot be turned back on without a reset of the chip. 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.

Read-modify-write operations on uninitialized RAM register produces undefined values.

Y.3.14 Application examples

Capacitive sense

Figure 259 illustrates how the ARM core can be configured to monitor four capacitive buttons.

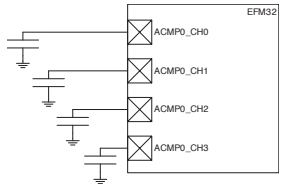


Figure 259: Capacitive sense setup

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 32kHz, set PCPRESC to 3 and PCTOP to 39 in CTRL. This will make the LESENSE scan frequency 100Hz.

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 (analog input).
3. Configure the ACMP to operate in CAPSENSE mode, refer to [?] for details.
4. Configure the following bit fields in CHx_CONF, for channels 0 through 3:
   1. Set EXTIME to 0. No excitation is needed in this mode.
   2. Set SAMPLE to COUNTER 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.
   3. Set SAMPLEDLY to an appropriate value, each sensor will be measured for SAMPLEDLY/$LFACLK_{LESENSE}$ 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, this is done in order 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.

LC sensor

Figure 260 below illustrates how the ARM core can be set up to monitor four LC sensors.

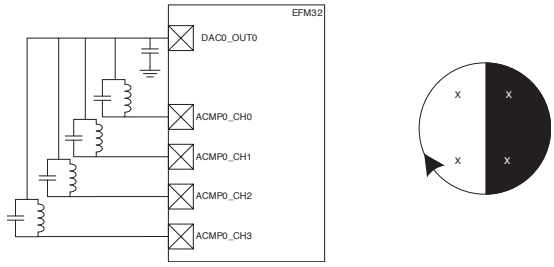


Figure 260: LC sensor setup

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 261 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.

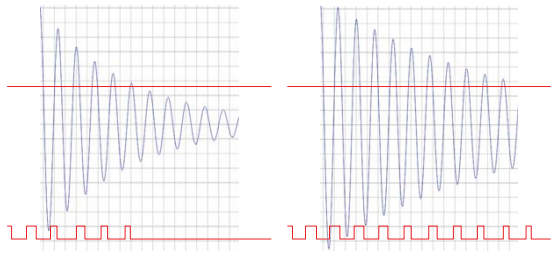


Figure 261: LC sensor oscillations

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 make the LESENSE scan frequency 100Hz.
2. Enable the DAC 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 set to Vdd/2 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:
   1. Set EXCLK to AUXHFRCO. AUXHFRCO is needed to achieve short excitation time.
   2. Set EXTIME to an appropriate value. Excitation will last for EXTIME/AUXHFRCO seconds (prescalar value in AUXPRESC in TIMCTRL is 0).
   3. Set EXMODE to LOW. The LC sensors are excited by pulling the excitation pin low.
   4. Set SAMPLE to COUNTER 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.
   5. Set SAMPLEDLY to an appropriate value, each sensor will be measured for SAMPLEDLY/$LFACLK_{LESENSE}$ 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 inactive, the counter value should be greater than the threshold.

9. Start scan sequence by writing a 1 to START in CMD.

LESENSE decoder 1

The example below illustrates how the LESENSE module can be used for decoding using three sensors

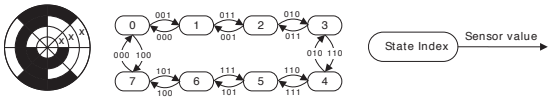


Figure 262: FSM example 1

To set up the decoder to decode rotation using the encoding scheme seen in Figure 262, 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:
   1. Set MASK = 0b1000 in STx_TCONFA and STx_TCONFB for all used states. This enables three sensors to be evaluated by the decoder.
   2. Configure the remaining bit fields in STx_TCONFA and STx_TCONFB as described in Table 263.

Register	TCONFA_NEXTSTATE	TCONFA_COMP	TCONFA_PRSACT	TCONFB_NEXTSTATE
ST0	1	0b001	UP	7
ST1	2	0b011	UP	0
ST2	3	0b010	UP	1
ST3	4	0b110	UP	2
ST4	5	0b111	UP	3
ST5	6	0b101	UP	4
ST6	7	0b100	UP	5
ST7	0	0b000	UP	6

Figure 263: LESENSE decoder configuration

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.

LESENSE decoder 2

The example below illustrates how the LESENSE decoder can be used to implement the state machine seen in Figure 264.

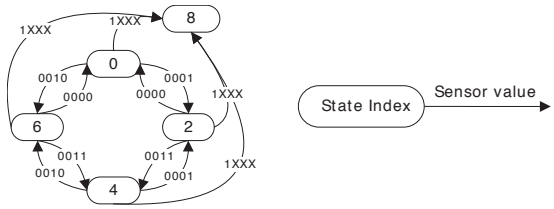


Figure 264: FSM example 2

1. Configure STx_TCONFA and STx_TCONFB as described in Table 265.

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	-

Figure 265: LESENSE decoder configuration

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.

Y.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_CMD	W1	Command Register
0x018	LESENSE_CHEN	RW	Channel enable Register
0x01C	LESENSE_SCANRES	R	Scan result register

Offset	Name	Type	Description
0x020	LESENSE_STATUS	R	Status Register
0x024	LESENSE_PTR	R	Result buffer pointers
0x028	LESENSE_BUFDATA	R	Result buffer data register
0x02C	LESENSE_CURCH	R	Current channel index
0x030	LESENSE_DECSTATE	RWH	Current decoder state
0x034	LESENSE_SENSORSTATE	RWH	Decoder input register
0x038	LESENSE_IDLECONF	RW	GPIO Idle phase configuration
0x03C	LESENSE_ALTEXCONF	RW	Alternative excite pin configuration
0x040	LESENSE_IF	R	Interrupt Flag Register
0x044	LESENSE_IFC	W1	Interrupt Flag Clear Register
0x048	LESENSE_IFS	W1	Interrupt Flag Set Register
0x04C	LESENSE_IEN	RW	Interrupt Enable Register
0x050	LESENSE_SYNCBUSY	R	Synchronization Busy Register
0x054	LESENSE_ROUTE	RW	I/O Routing Register
0x058	LESENSE_POWERDOWN	RW	LESENSE RAM power-down register
0x200	LESENSE_ST0_TCONFA	RW	State transition configuration A
0x204	LESENSE_ST0_TCONFB	RW	State transition configuration B
...	LESENSE_STx_TCONFA	RW	State transition configuration A
...	LESENSE_STx_TCONFB	RW	State transition configuration B
0x278	LESENSE_ST15_TCONFA	RW	State transition configuration A
0x27C	LESENSE_ST15_TCONFB	RW	State transition configuration B
0x280	LESENSE_BUF0_DATA	RW	Scan results
...	LESENSE_BUFx_DATA	RW	Scan results
0x2BC	LESENSE_BUF15_DATA	RW	Scan results
0x2C0	LESENSE_CH0_TIMING	RW	Scan configuration
0x2C4	LESENSE_CH0_INTERACT	RW	Scan configuration
0x2C8	LESENSE_CH0_EVAL	RW	Scan configuration
...	LESENSE_CHx_TIMING	RW	Scan configuration
...	LESENSE_CHx_INTERACT	RW	Scan configuration
...	LESENSE_CHx_EVAL	RW	Scan configuration
0x3B0	LESENSE_CH15_TIMING	RW	Scan configuration
0x3B4	LESENSE_CH15_INTERACT	RW	Scan configuration

Offset	Name	Type	Description
0x3B8	LESENSE_CH15_EVAL	RW	Scan configuration

Y.5 Register Description

Y.5.1 LESENSE_CTRL - Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	0	0		0x0		0x0				0x0	
Access										RW	RW			RW	RW	RW			RW		RW	RW	RW		RW		RW				RW	
Name										DEBUGRUN	DMAWU			BUFIDL	STRSCANRES	BUFOW			DUALSAMPLE		ALTEXMAP	ACMP1INV	ACMP0INV		SCANCONF		PRSSEL				SCANMODE	

Bit	Name	Reset	Access	Description
31:23	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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

Bit	Name	Reset	Access	Description
21:20	DMAWU	0x0	RW	**DMA wake-up from 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

Bit	Name	Reset	Access	Description
19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18	BUFIDL	0	RW	**Result buffer interrupt and DMA trigger level**

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

Bit	Name	Reset	Access	Description
17	STRSCANRES	0	RW	**Enable storing of SCANRES**
	When set, SCANRES will be stored in the result buffer after each scan			
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.		
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.		
11	ALTEXMAP	0	RW	**Alternative excitation map**

Value	Mode	Description
0	ALTEX	Alternative excitation is mapped to the LES_ALTEX pins.
1	ACMP	Alternative excitation is mapped to the pins of the other ACMP.

Bit	Name	Reset	Access	Description
10	ACMP1INV	0	RW	**Invert analog comparator 1 output**

Bit	Name	Reset	Access	Description
9	ACMP0INV	0	RW	**Invert analog comparator 0 output**
8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:6	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.

Bit	Name	Reset	Access	Description
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
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
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

Y.5.2 LESENSE_TIMCTRL - Timing Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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				0x0					0x0	
Access									RW				RW									RW				RW					RW	
Name									STARTDLY				PCTOP									PCPRESC				LFPRESC					AUXPRESC	

Bit	Name	Reset	Access	Description
31:24	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.		

Bit	Name	Reset	Access	Description
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.		
10:8	PCPRESC	0x0	RW	**Period counter prescaling**

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
6	DIV64	The period counter clock frequency is $LFACLK_{LESENSE}$/64
7	DIV128	The period counter clock frequency is $LFACLK_{LESENSE}$/128

Bit	Name	Reset	Access	Description
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:4	LFPRESC	0x0	RW	**Prescaling factor for 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

Bit	Name	Reset	Access	Description
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1:0	AUXPRESC	0x0	RW	**Prescaling factor for 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

Y.5.3 LESENSE_PERCTRL - Peripheral Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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			0				0x00					0x0		0x0		0x0		0x0		0	0
Access					RW				RW		RW			RW				RW					RW		RW		RW		RW		RW	RW
Name					WARMUPMODE				ACMP1MODE		ACMP0MODE			DACREF				DACPRESC					DACCH1OUT		DACCH0OUT		DACCH1CONV		DACCH0CONV		DACCH1DATA	DACCH0DATA

Bit	Name	Reset	Access	Description
31:28	Reserved	To ensure compatibility with future devices, always write bits to 0.		
27:26	WARMUPMODE	0x0	RW	**ACMP and DAC duty cycle mode**

Bit	Name	Reset	Access	Description
	Value	**Mode**		**Description**
	0	NORMAL		The analog comparators and DAC are shut down when LESENSE is idle
	1	KEEPACMPWARM		The analog comparators are kept powered up when LESENSE is idle
	2	KEEPDACWARM		The DAC is kept powered up when LESENSE is idle
	3	KEEPACMPDACWARM		The analog comparators and DAC are kept powered up when LESENSE is idle
25:24	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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
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	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18	DACREF	0	RW	**DAC bandgap reference used**
	Set to BANDGAP if the DAC is configured to use bandgap reference			
	Value	**Mode**		**Description**
	0	VDD		DAC uses VDD reference
	1	BANDGAP		DAC uses bandgap reference
17:15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14:10	DACPRESC	0x00	RW	**DAC prescaler configuration.**
	Prescaling factor of DACPRESC+1 for the LESENSE DAC interface			
9:8	DACCH1OUT	0x0	RW	**DAC channel 1 output mode**
	Value	**Mode**		**Description**
	0	DISABLE		DAC CH1 output to pin and ACMP/ADC disabled
	1	PIN		DAC CH1 output to pin enabled, output to ADC and ACMP disabled
	2	ADCACMP		DAC CH1 output to pin disabled, output to ADC and ACMP enabled
	3	PINADCACMP		DAC CH1 output to pin, ADC, and ACMP enabled.
7:6	DACCH0OUT	0x0	RW	**DAC channel 0 output mode**
	Value	**Mode**		**Description**
	0	DISABLE		DAC CH0 output to pin and ACMP/ADC disabled
	1	PIN		DAC CH0 output to pin enabled, output to ADC and ACMP disabled
	2	ADCACMP		DAC CH0 output to pin disabled, output to ADC and ACMP enabled
	3	PINADCACMP		DAC CH0 output to pin, ADC, and ACMP enabled.
5:4	DACCH1CONV	0x0	RW	**DAC channel 1 conversion mode**
	Value	**Mode**		**Description**
	0	DISABLE		LESENSE does not control DAC CH1.
	1	CONTINUOUS		DAC channel 1 is driven in continuous mode.
	2	SAMPLEHOLD		DAC channel 1 is driven in sample hold mode.
	3	SAMPLEOFF		DAC channel 1 is driven in sample off mode.
3:2	DACCH0CONV	0x0	RW	**DAC channel 0 conversion mode**

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	DISABLE	LESENSE does not control DAC CH0.
1	CONTINUOUS	DAC channel 0 is driven in continuous mode.
2	SAMPLEHOLD	DAC channel 0 is driven in sample hold mode.
3	SAMPLEOFF	DAC channel 0 is driven in sample off mode.

Bit	Name	Reset	Access	Description
1	DACCH1DATA	0	RW	**DAC CH1 data selection.**

Configure DAC data control.

Value	Mode	Description
0	DACDATA	DAC data is defined by CH1DATA in the DAC interface.
1	ACMPTHRES	DAC data is defined by ACMPTHRES in CHx_INTERACT.

Bit	Name	Reset	Access	Description
0	DACCH0DATA	0	RW	**DAC CH0 data selection.**

Value	Mode	Description
0	DACDATA	DAC data is defined by CH0DATA in the DAC interface.
1	ACMPTHRES	DAC data is defined by ACMPTHRES in CHx_INTERACT.

Y.5.4 LESENSE_DECCTRL - Decoder control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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
31:26	Reserved	To ensure compatibility with future devices, always write bits to 0.		
25:22	PRSSEL3	0x0	RW	

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

Bit	Name	Reset	Access	Description
21:18	PRSSEL2	0x0	RW	

Select PRS input for bit 2 of the LESENSE decoder

Bit	Name	Reset	Access	Description

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

17:14 PRSSEL1 0x0 RW

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

13:10 PRSSEL0 0x0 RW

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

9 Reserved To ensure compatibility with future devices, always write bits to 0.

8 INPUT 0 RW

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

Bit	Name	Reset	Access	Description
4	HYSTPRS1	0	RW	**Enable decoder hysteresis on PRS1 output**
	When set, hysteresis is enabled in the decoder, suppressing changes on PRS channel 1			
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 CHx			
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			

Y.5.5 LESENSE_BIASCTRL - Bias Control Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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.		
1:0	BIASMODE	0x0	RW	**Select bias mode**

Value	Mode	Description
0	DUTYCYCLE	Bias module duty cycled between low power and high accuracy mode
1	HIGHACC	Bias module always in high accuracy mode
2	DONTTOUCH	Bias module is controlled by the EMU and not affected by LESENSE

Y.5.6 LESENSE_CMD - Command Register

Offset	Bit Position																													
0x014	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																													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.		
3	CLEARBUF	0	W1	**Clear result buffer**
2	DECODE	0	W1	**Start decoder**

Bit	Name	Reset	Access	Description
1	STOP	0	W1	**Stop scanning of sensors**
	If issued during a scan, the command will take effect after scan completion.			
0	START	0	W1	**Start scanning of sensors.**

Y.5.7 LESENSE_CHEN - Channel enable Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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																	RW															
Name																	CHEN															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	CHEN	0x0000	RW	**Enable scan channel**
	Set bit X to enable channel X			

Y.5.8 LESENSE_SCANRES - Scan result register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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																	SCANRES															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	SCANRES	0x0000	R	**Scan results**
	Bit X will be set depending on channel X evaluation			

Y.5.9 LESENSE_STATUS - Status Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	R	R	R	R	R
Name																											DACACTIVE	SCANACTIVE	RUNNING	BUFFULL	BUFHALFULL	BUFDATAV

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5	DACACTIVE	0	R	**LESENSE DAC interface is active**
4	SCANACTIVE	0	R	**LESENSE is currently interfacing sensors.**
3	RUNNING	0	R	**LESENSE is active**
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.			

Y.5.10 LESENSE_PTR - Result buffer pointers (Async Reg)

For more information about Asynchronous Registers please see F.4.

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

Bit	Name	Reset	Access	Description
31:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8:5	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			
4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

Y.5.11 LESENSE_BUFDATA - Result buffer data register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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																	0xXXXX															
Access																	R															
Name																	BUFDATA															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	BUFDATA	0xXXXX	R	**Result data**
	This register can be used to read the oldest unread data from the result buffer.			

Y.5.12 LESENSE_CURCH - Current channel index (Async Reg)

For more information about Asynchronous Registers please see F.4.

Offset	Bit Position																															
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	R																															
Name	CURCH																															

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3:0	CURCH	0x0	R	**Shows the index of the current channel**

Y.5.13 LESENSE_DECSTATE - Current decoder state (Async Reg)

For more information about Asynchronous Registers please see F.4.

Offset	Bit Position																															
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0																															
Access	RWH																															
Name	DECSTATE																															

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3:0	DECSTATE	0x0	RWH	**Shows the current decoder state**

Y.5.14 LESENSE_SENSORSTATE - Decoder input register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	RWH																															
Name	SENSORSTATE																															

Bit	Name	Reset	Access	Description
31:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3:0	SENSORSTATE	0x0	RWH	**Shows the status of sensors chosen as input to the decoder**

Y.5.15 LESENSE_IDLECONF - GPIO Idle phase configuration (Async Reg)

For more information about Asynchronous Registers please see F.4.

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

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	DACCH1	CH15 output is connected to DAC CH1 output in idle phase

Bit	Name	Reset	Access	Description
29:28	CH14	0x0	RW	**Channel 14 idle phase configuration**

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	DACCH1	CH14 output is connected to DAC CH1 output in idle phase

Bit	Name	Reset	Access	Description
27:26	CH13	0x0	RW	**Channel 13 idle phase configuration**

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	DACCH1	CH13 output is connected to DAC CH1 output in idle phase

Bit	Name	Reset	Access	Description
25:24	CH12	0x0	RW	**Channel 12 idle phase configuration**

Value	Mode	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	DACCH1	CH12 output is connected to DAC CH1 output in idle phase

Bit	Name	Reset	Access	Description
23:22	CH11	0x0	RW	**Channel 11 idle phase configuration**

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

Bit	Name	Reset	Access	Description
21:20	CH10	0x0	RW	**Channel 10 idle phase configuration**

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

Bit	Name	Reset	Access	Description
19:18	CH9	0x0	RW	**Channel 9 idle phase configuration**

Bit	Name	Reset	Access	Description

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

17:16 CH8 0x0 RW **Channel 8 idle phase configuration**

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

15:14 CH7 0x0 RW **Channel 7 idle phase configuration**

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

13:12 CH6 0x0 RW **Channel 6 idle phase configuration**

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

11:10 CH5 0x0 RW **Channel 5 idle phase configuration**

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

9:8 CH4 0x0 RW **Channel 4 idle phase configuration**

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

7:6 CH3 0x0 RW **Channel 3 idle phase configuration**

Value	Mode	Description
0	DISABLE	CH3 output is disabled in idle phase
1	HIGH	CH3 output is high in idle phase
2	LOW	CH3 output is low in idle phase
3	DACCH0	CH3 output is connected to DAC CH0 output in idle phase

5:4 CH2 0x0 RW **Channel 2 idle phase configuration**

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	DACCH0	CH2 output is connected to DAC CH0 output in idle phase

3:2 CH1 0x0 RW **Channel 1 idle phase configuration**

Bit	Name	Reset	Access	Description
	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	DACCH0		CH1 output is connected to DAC CH0 output in idle phase
1:0	CH0	0x0	RW	**Channel 0 idle phase configuration**
	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	DACCH0		CH0 output is connected to DAC CH0 output in idle phase

Y.5.16 LESENSE_ALTEXCONF - Alternative excite pin configuration (Async Reg)

For more information about Asynchronous Registers please see F.4.

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	0x0		0x0		0x0		0x0		0x0		0x0		0x0	
Access									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.		
23	AEX7	0	RW	**ALTEX7 always excite enable**
22	AEX6	0	RW	**ALTEX6 always excite enable**
21	AEX5	0	RW	**ALTEX5 always excite enable**
20	AEX4	0	RW	**ALTEX4 always excite enable**
19	AEX3	0	RW	**ALTEX3 always excite enable**
18	AEX2	0	RW	**ALTEX2 always excite enable**
17	AEX1	0	RW	**ALTEX1 always excite enable**
16	AEX0	0	RW	**ALTEX0 always excite enable**
15:14	IDLECONF7	0x0	RW	**ALTEX7 idle phase configuration**
	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**

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**
	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**
	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**
	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**
	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**
	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
1:0	IDLECONF0	0x0	RW	**ALTEX0 idle phase configuration**
	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

Y.5.17 LESENSE_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	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.		
22	CNTOF	0	R	
	Set when the LESENSE counter overflows.			
21	BUFOF	0	R	
	Set when the result buffer overflows			
20	BUFLEVEL	0	R	
	Set when the data buffer is full.			
19	BUFDATAV	0	R	
	Set when data is available in the result buffer.			
18	DECERR	0	R	
	Set when the decoder detects an error			
17	DEC	0	R	
	Set when the decoder has issued and interrupt request			
16	SCANCOMPLETE	0	R	
	Set when a scan sequence is completed			
15	CH15	0	R	
	Set when channel 15 triggers			
14	CH14	0	R	
	Set when channel 14 triggers			
13	CH13	0	R	
	Set when channel 13 triggers			
12	CH12	0	R	
	Set when channel 12 triggers			
11	CH11	0	R	
	Set when channel 11 triggers			
10	CH10	0	R	
	Set when channel 10 triggers			
9	CH9	0	R	
	Set when channel 9 triggers			
8	CH8	0	R	
	Set when channel 8 triggers			
7	CH7	0	R	
	Set when channel 7 triggers			
6	CH6	0	R	
	Set when channel 6 triggers			
5	CH5	0	R	
	Set when channel 5 triggers			
4	CH4	0	R	
	Set when channel 4 triggers			
3	CH3	0	R	
	Set when channel 3 triggers			
2	CH2	0	R	
	Set when channel 2 triggers			
1	CH1	0	R	
	Set when channel 1 triggers			
0	CH0	0	R	
	Set when channel 0 triggers			

Y.5.18 LESENSE_IFC - Interrupt Flag Clear 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	0	0	0	0	0	0	0	0

Offset	Bit Position																								
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
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.		
22	CNTOF	0	W1	
	Write to 1 to clear CNTOF interrupt flag			
21	BUFOF	0	W1	
	Write to 1 to clear BUFOF interrupt flag			
20	BUFLEVEL	0	W1	
	Write to 1 to clear BUFLEVEL interrupt flag			
19	BUFDATAV	0	W1	
	Write to 1 to clear BUFDATAV interrupt flag			
18	DECERR	0	W1	
	Write to 1 to clear DECERR interrupt flag			
17	DEC	0	W1	
	Write to 1 to clear DEC interrupt flag			
16	SCANCOMPLETE	0	W1	
	Write to 1 to clear SCANCOMPLETE interrupt flag			
15	CH15	0	W1	
	Write to 1 to clear CH15 interrupt flag			
14	CH14	0	W1	
	Write to 1 to clear CH14 interrupt flag			
13	CH13	0	W1	
	Write to 1 to clear CH13 interrupt flag			
12	CH12	0	W1	
	Write to 1 to clear CH12 interrupt flag			
11	CH11	0	W1	
	Write to 1 to clear CH11 interrupt flag			
10	CH10	0	W1	
	Write to 1 to clear CH10 interrupt flag			
9	CH9	0	W1	
	Write to 1 to clear CH9 interrupt flag			
8	CH8	0	W1	
	Write to 1 to clear CH8 interrupt flag			
7	CH7	0	W1	
	Write to 1 to clear CH7 interrupt flag			
6	CH6	0	W1	
	Write to 1 to clear CH6 interrupt flag			
5	CH5	0	W1	
	Write to 1 to clear CH5 interrupt flag			
4	CH4	0	W1	
	Write to 1 to clear CH4 interrupt flag			
3	CH3	0	W1	
	Write to 1 to clear CH3 interrupt flag			
2	CH2	0	W1	
	Write to 1 to clear CH2 interrupt flag			
1	CH1	0	W1	
	Write to 1 to clear CH1 interrupt flag			
0	CH0	0	W1	

Bit	Name	Reset	Access	Description
	Write to 1 to clear CH0 interrupt flag			

Y.5.19 LESENSE_IFS - Interrupt Flag Set 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	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.		
22	CNTOF	0	W1	
	Write to 1 to set the CNTOF interrupt flag			
21	BUFOF	0	W1	
	Write to 1 to set the BUFOF interrupt flag			
20	BUFLEVEL	0	W1	
	Write to 1 to set the BUFLEVEL interrupt flag			
19	BUFDATAV	0	W1	
	Write to 1 to set the BUFDATAV interrupt flag			
18	DECERR	0	W1	
	Write to 1 to set the DECERR interrupt flag			
17	DEC	0	W1	
	Write to 1 to set the DEC interrupt flag			
16	SCANCOMPLETE	0	W1	
	Write to 1 to set the SCANCOMPLETE interrupt flag			
15	CH15	0	W1	
	Write to 1 to set the CH15 interrupt flag			
14	CH14	0	W1	
	Write to 1 to set the CH14 interrupt flag			
13	CH13	0	W1	
	Write to 1 to set the CH13 interrupt flag			
12	CH12	0	W1	
	Write to 1 to set the CH12 interrupt flag			
11	CH11	0	W1	
	Write to 1 to set the CH11 interrupt flag			
10	CH10	0	W1	
	Write to 1 to set the CH10 interrupt flag			
9	CH9	0	W1	
	Write to 1 to set the CH9 interrupt flag			
8	CH8	0	W1	
	Write to 1 to set the CH8 interrupt flag			
7	CH7	0	W1	
	Write to 1 to set the CH7 interrupt flag			
6	CH6	0	W1	
	Write to 1 to set the CH6 interrupt flag			
5	CH5	0	W1	
	Write to 1 to set the CH5 interrupt flag			

Bit	Name	Reset	Access	Description
4	CH4	0	W1	
	Write to 1 to set the CH4 interrupt flag			
3	CH3	0	W1	
	Write to 1 to set the CH3 interrupt flag			
2	CH2	0	W1	
	Write to 1 to set the CH2 interrupt flag			
1	CH1	0	W1	
	Write to 1 to set the CH1 interrupt flag			
0	CH0	0	W1	
	Write to 1 to set the CH0 interrupt flag			

Y.5.20 LESENSE_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	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.		
22	CNTOF	0	RW	
	Set to enable interrupt on the CNTOF interrupt flag			
21	BUFOF	0	RW	
	Set to enable interrupt on the BUFOF interrupt flag			
20	BUFLEVEL	0	RW	
	Set to enable interrupt on the BUFLEVEL interrupt flag			
19	BUFDATAV	0	RW	
	Set to enable interrupt on the BUFDATAV interrupt flag			
18	DECERR	0	RW	
	Set to enable interrupt on the DECERR interrupt flag			
17	DEC	0	RW	
	Set to enable interrupt on the DEC interrupt flag			
16	SCANCOMPLETE	0	RW	
	Set to enable interrupt on the SCANCOMPLETE interrupt flag			
15	CH15	0	RW	
	Set to enable interrupt on the CH15 interrupt flag			
14	CH14	0	RW	
	Set to enable interrupt on the CH14 interrupt flag			
13	CH13	0	RW	
	Set to enable interrupt on the CH13 interrupt flag			
12	CH12	0	RW	
	Set to enable interrupt on the CH12 interrupt flag			
11	CH11	0	RW	
	Set to enable interrupt on the CH11 interrupt flag			
10	CH10	0	RW	
	Set to enable interrupt on the CH10 interrupt flag			
9	CH9	0	RW	

Bit	Name	Reset	Access	Description
	Set to enable interrupt on the CH9 interrupt flag			
8	CH8	0	RW	
	Set to enable interrupt on the CH8 interrupt flag			
7	CH7	0	RW	
	Set to enable interrupt on the CH7 interrupt flag			
6	CH6	0	RW	
	Set to enable interrupt on the CH6 interrupt flag			
5	CH5	0	RW	
	Set to enable interrupt on the CH5 interrupt flag			
4	CH4	0	RW	
	Set to enable interrupt on the CH4 interrupt flag			
3	CH3	0	RW	
	Set to enable interrupt on the CH3 interrupt flag			
2	CH2	0	RW	
	Set to enable interrupt on the CH2 interrupt flag			
1	CH1	0	RW	
	Set to enable interrupt on the CH1 interrupt flag			
0	CH0	0	RW	
	Set to enable interrupt on the CH0 interrupt flag			

Y.5.21 LESENSE_SYNCBUSY - Synchronization Busy 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	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
Name						EVAL	INTERACT	TIMING	DATA	TCONFB	TCONFA				POWERDOWN	ROUTE	ALTEXCONF	IDLECONF	SENSORSTATE	DECSTATE	CURCH	BUFDATA	PTR	STATUS	SCANRES	CHEN	CMD	BIASCTRL	DECCTRL	PERCTRL	TIMCTRL	CTRL

Bit	Name	Reset	Access	Description
31:27	Reserved	To ensure compatibility with future devices, always write bits to 0.		
26	EVAL	0	R	**LESENSE_CHx_EVAL Register Busy**
	Set when the value written to LESENSE_CHx_EVAL is being synchronized.			
25	INTERACT	0	R	**LESENSE_CHx_INTERACT Register Busy**
	Set when the value written to LESENSE_CHx_INTERACT is being synchronized.			
24	TIMING	0	R	**LESENSE_CHx_TIMING Register Busy**
	Set when the value written to LESENSE_CHx_TIMING is being synchronized.			
23	DATA	0	R	**LESENSE_BUFx_DATA Register Busy**
	Set when the value written to LESENSE_BUFx_DATA is being synchronized.			
22	TCONFB	0	R	**LESENSE_STx_TCONFB Register Busy**
	Set when the value written to LESENSE_STx_TCONFB is being synchronized.			
21	TCONFA	0	R	**LESENSE_STx_TCONFA Register Busy**
	Set when the value written to LESENSE_STx_TCONFA is being synchronized.			
20:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17	POWERDOWN	0	R	**LESENSE_POWERDOWN Register Busy**
	Set when the value written to LESENSE_POWERDOWN is being synchronized.			
16	ROUTE	0	R	**LESENSE_ROUTE Register Busy**
	Set when the value written to LESENSE_ROUTE is being synchronized.			
15	ALTEXCONF	0	R	**LESENSE_ALTEXCONF Register Busy**
	Set when the value written to LESENSE_ALTEXCONF is being synchronized.			

Bit	Name	Reset	Access	Description
14	IDLECONF	0	R	**LESENSE_IDLECONF Register Busy**
	Set when the value written to LESENSE_IDLECONF is being synchronized.			
13	SENSORSTATE	0	R	**LESENSE_SENSORSTATE Register Busy**
	Set when the value written to LESENSE_SENSORSTATE is being synchronized.			
12	DECSTATE	0	R	**LESENSE_DECSTATE Register Busy**
	Set when the value written to LESENSE_DECSTATE is being synchronized.			
11	CURCH	0	R	**LESENSE_CURCH Register Busy**
	Set when the value written to LESENSE_CURCH is being synchronized.			
10	BUFDATA	0	R	**LESENSE_BUFDATA Register Busy**
	Set when the value written to LESENSE_BUFDATA is being synchronized.			
9	PTR	0	R	**LESENSE_PTR Register Busy**
	Set when the value written to LESENSE_PTR is being synchronized.			
8	STATUS	0	R	**LESENSE_STATUS Register Busy**
	Set when the value written to LESENSE_STATUS is being synchronized.			
7	SCANRES	0	R	**LESENSE_SCANRES Register Busy**
	Set when the value written to LESENSE_SCANRES is being synchronized.			
6	CHEN	0	R	**LESENSE_CHEN Register Busy**
	Set when the value written to LESENSE_CHEN is being synchronized.			
5	CMD	0	R	**LESENSE_CMD Register Busy**
	Set when the value written to LESENSE_CMD is being synchronized.			
4	BIASCTRL	0	R	**LESENSE_BIASCTRL Register Busy**
	Set when the value written to LESENSE_BIASCTRL is being synchronized.			
3	DECCTRL	0	R	**LESENSE_DECCTRL Register Busy**
	Set when the value written to LESENSE_DECCTRL is being synchronized.			
2	PERCTRL	0	R	**LESENSE_PERCTRL Register Busy**
	Set when the value written to LESENSE_PERCTRL is being synchronized.			
1	TIMCTRL	0	R	**LESENSE_TIMCTRL Register Busy**
	Set when the value written to LESENSE_TIMCTRL is being synchronized.			
0	CTRL	0	R	**LESENSE_CTRL Register Busy**
	Set when the value written to LESENSE_CTRL is being synchronized.			

Y.5.22 LESENSE_ROUTE - I/O Routing Register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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
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.		
23	ALTEX7PEN	0	RW	**ALTEX7 Pin Enable**
22	ALTEX6PEN	0	RW	**ALTEX6 Pin Enable**
21	ALTEX5PEN	0	RW	**ALTEX5 Pin Enable**
20	ALTEX4PEN	0	RW	**ALTEX4 Pin Enable**

Bit	Name	Reset	Access	Description
19	ALTEX3PEN	0	RW	**ALTEX3 Pin Enable**
18	ALTEX2PEN	0	RW	**ALTEX2 Pin Enable**
17	ALTEX1PEN	0	RW	**ALTEX1 Pin Enable**
16	ALTEX0PEN	0	RW	**ALTEX0 Pin Enable**
15	CH15PEN	0	RW	**CH15 Pin Enable**
14	CH14PEN	0	RW	**CH14 Pin Enable**
13	CH13PEN	0	RW	**CH13 Pin Enable**
12	CH12PEN	0	RW	**CH12 Pin Enable**
11	CH11PEN	0	RW	**CH11 Pin Enable**
10	CH10PEN	0	RW	**CH10 Pin Enable**
9	CH9PEN	0	RW	**CH9 Pin Enable**
8	CH8PEN	0	RW	**CH8 Pin Enable**
7	CH7PEN	0	RW	**CH7 Pin Enable**
6	CH6PEN	0	RW	**CH6 Pin Enable**
5	CH5PEN	0	RW	**CH5 Pin Enable**
4	CH4PEN	0	RW	**CH4 Pin Enable**
3	CH3PEN	0	RW	**CH3 Pin Enable**
2	CH2PEN	0	RW	**CH2 Pin Enable**
1	CH1PEN	0	RW	**CH0 Pin Enable**
0	CH0PEN	0	RW	**CH0 Pin Enable**

Y.5.23 LESENSE_POWERDOWN - LESENSE RAM power-down register (Async Reg)

For more information about Asynchronous Registers please see F.4.

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
Access																																RW
Name																																RAM

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	RAM	0	RW	**LESENSE RAM power-down**
	Shut off power to the LESENSE RAM. Once it is powered down, it cannot be powered up again			

Y.5.24 LESENSE_STx_TCONFA - State transition configuration A (Async Reg)

For more information about Asynchronous Registers please see F.4.

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														X		X		0xX			0xX				0xX				0xX		
Access														RW		RW		RW			RW				RW				RW		
Name														CHAIN		SETIF		PRSACT			NEXTSTATE				MASK				COMP		

Bit	Name	Reset	Access	Description
31:19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18	CHAIN	X	RW	**Enable state descriptor chaining**
	When set, descriptor in the next location will also be evaluated			
17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	SETIF	X	RW	**Set interrupt flag enable**
	Set interrupt flag when sensor state equals COMP			
15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14:12	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.

11:8	NEXTSTATE	0xX	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			

Y.5.25 LESENSE_STx_TCONFB - State transition configuration B (Async Reg)

For more information about Asynchronous Registers please see F.4.

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
Reset																X		0xX			0xX				0xX				0xX			
Access																RW		RW			RW				RW				RW			
Name																SETIF		PRSACT			NEXTSTATE				MASK				COMP			

Bit	Name	Reset	Access	Description
31:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	SETIF	X	RW	**Set interrupt flag**
	Set interrupt flag when sensor state equals COMP			
15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14:12	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.

11:8	NEXTSTATE	0xX	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			

Y.5.26 LESENSE_BUFx_DATA - Scan results (Async Reg)

For more information about Asynchronous Registers please see F.4.

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

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	DATA	0xXXXX	RW	**Scan result buffer**

Y.5.27 LESENSE_CHx_TIMING - Scan configuration (Async Reg)

For more information about Asynchronous Registers please see F.4.

Offset	Bit Position																														
0x2C0	31	30	29	28	27	26	25	24	23	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							0xXX							0xXX					
Access													RW							RW							RW					
Name													MEASUREDLY							SAMPLEDLY							EXTIME					

Bit	Name	Reset	Access	Description
31:20	Reserved	To ensure compatibility with future devices, always write bits to 0.		
19:13	MEASUREDLY	0xXX	RW	**Set measure delay**
	Configure measure delay. Sensor measuring is delayed for MEASUREDLY+1 EXCLK cycles.			
12:6	SAMPLEDLY	0xXX	RW	**Set sample delay**
	Configure sample delay. Sampling will occur after SAMPLEDLY+1 SAMPLECLK cycles.			
5:0	EXTIME	0xXX	RW	**Set excitation time**
	Configure excitation time. Excitation will last EXTIME+1 EXCLK cycles.			

Y.5.28 LESENSE_CHx_INTERACT - Scan configuration (Async Reg)

For more information about Asynchronous Registers please see F.4.

Offset	Bit Position																														
0x2C4	31	30	29	28	27	26	25	24	23	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		X	0xXXX											
Access													RW	RW	RW	RW		RW		RW	RW											
Name													ALTEX	SAMPLECLK	EXCLK	EXMODE		SETIF		SAMPLE	ACMPTHRES											

Bit	Name	Reset	Access	Description
31:20	Reserved	To ensure compatibility with future devices, always write bits to 0.		
19	ALTEX	X	RW	**Use alternative excite pin**
	If set, alternative excite pin will be used for excitation			
18	SAMPLECLK	X	RW	**Select clock used for timing of sample delay**

Value	Mode	Description
0	LFACLK	LFACLK will be used for timing
1	AUXHFRCO	AUXHFRCO will be used for timing

17	EXCLK	X	RW	**Select clock used for excitation timing**

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	LFACLK	LFACLK will be used for timing
1	AUXHFRCO	AUXHFRCO will be used for timing

Bit	Name	Reset	Access	Description
16:15	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, and 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	DAC output

Bit	Name	Reset	Access	Description
14:13	SETIF	0xX	RW	**Enable interrupt generation**

Select interrupt generation mode for CHx interrupt flag.

Value	Mode	Description
0	NONE	No interrupt is generated
1	LEVEL	Set interrupt flag if the sensor triggers.
2	POSEDGE	Set interrupt flag on positive edge on the sensor state
3	NEGEDGE	Set interrupt flag on negative edge on the sensor state

Bit	Name	Reset	Access	Description
12	SAMPLE	X	RW	**Select sample mode**

Select if ACMP output or counter output should be used in comparison

Value	Mode	Description
0	COUNTER	Counter output will be used in comparison
1	ACMP	ACMP output will be used in comparison

Bit	Name	Reset	Access	Description
11:0	ACMPTHRES	0xXXX	RW	**Set ACMP threshold**

Select ACMP threshold.

Y.5.29 LESENSE_CHx_EVAL - Scan configuration (Async Reg)

For more information about Asynchronous Registers please see F.4.

Offset	Bit Position																															
0x2C8	31	30	29	28	27	26	25	24	23	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	X	0xXXXX															
Access													RW	RW	RW	RW	RW															
Name													SCANRESINV	STRSAMPLE	DECODE	COMP	COMPTHRES															

Bit	Name	Reset	Access	Description
31:20	Reserved	To ensure compatibility with future devices, always write bits to 0.		
19	SCANRESINV	X	RW	**Enable inversion of result**

If set, the bit stored in SCANRES will be inverted.

Bit	Name	Reset	Access	Description
18	STRSAMPLE	X	RW	**Select if counter result should be stored**

If set, the counter value will be stored and available in the result buffer

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 counter comparison**

Set compare mode

Bit	Name	Reset	Access	Description
	CH_INTERACT_SAMPLE = COUNTER			
	Mode	Value		Description
	LESS	0		Comparison evaluates to 1 if counter value is less than COMPTHRES.
	GE	1		Comparison evaluates to 1 if counter value is greater than, or equal to COMPTHRES.
	CH_INTERACT_SAMPLE = ACMP			
	LESS	0		Comparison evaluates to 1 if the ACMP output is 0.
	GE	1		Comparison evaluates to 1 if the ACMP output is 1.
15:0	COMPTHRES	0xXXXX	RW	**Decision threshold for counter**
	Set counter threshold			

Z ARM Analog Comparator

Z.1 Introduction

The Analog Comparator is used to compare the voltage of two analog inputs, with a digital output indicating which input voltage is higher. Inputs can either be one of the selectable internal references or from external pins. Response time and thereby also the current consumption can be configured by altering the current supply to the comparator.

Z.2 Features

- 8 selectable external positive inputs
- 8 selectable external negative inputs
- 5 selectable internal negative inputs
  - Internal 1.25 V bandgap
  - Internal 2.5 V bandgap
  - V_{DD} scaled by 64 selectable factors
  - DAC channel 0 and 1
- Low power mode for internal V_{DD} and bandgap references
- Selectable hysteresis
  - 8 levels between 0 and $\pm$70 mV
- Selectable response time
- Asynchronous interrupt generation on selectable edges
  - Rising edge
  - Falling edge
  - Both edges
- Operational in EM0-EM3
- Dedicated capacitive sense mode with up to 8 inputs
  - Adjustable internal resistor
- Configurable inversion of comparator output
- Configurable output when inactive
- Comparator output direct on PRS
- Comparator output on GPIO through alternate functionality
  - Output inversion available

Z.3 Functional Description

An overview of the ACMP is shown in Figure 266.

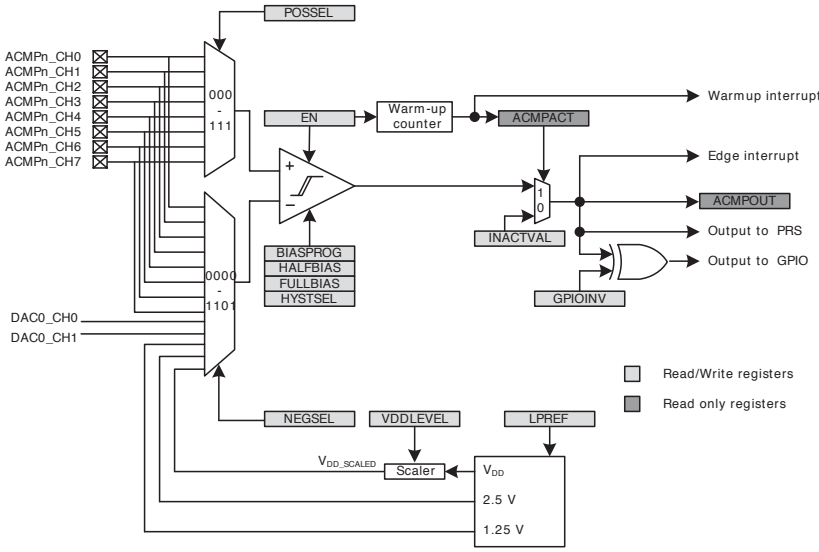


Figure 266: ACMP Overview

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.

Z.3.1 Warm-up Time

The analog comparator is enabled by setting the EN bit in ACMPn_CTRL. When this bit is set, the comparator must stabilize before becoming active and the outputs can be used. This time period is called the warm-up time. The warm-up time is a configurable number of peripheral clock (HFPERCLK) cycles, set in WARMTIME, which should be set to at least 10 μs but lengthens to up to 1ms if LPREF is enabled. The ACMP should always start in active mode and then enable the LPREF after warm-up time. When the comparator is enabled and warmed up, the ACMPACT bit in ACMPn_STATUS will indicate that the comparator is active. The output value when the comparator is inactive is set to the value in INACTVAL in ACMPn_CTRL (see Figure 266).

An edge interrupt will be generated after the warm-up time if edge interrupt is enabled and the value set in INACTVAL is different from ACMPOUT after warm-up.

One 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.

Z.3.2 Response Time

There is a delay from when the actual input voltage changes polarity, to when the output toggles. This period is called the response time and can be altered by increasing or decreasing the bias current to the comparator through the BIASPROG, FULLBIASPROG and HALFBIAS fields in the ACMPn_CTRL register, as illustrated in Table 267 Setting the HALFBIAS bit in ACMPn_CTRL effectively halves the current. Setting a lower bias current will result in lower power consumption, but a longer response time.

If the FULLBIAS bit is set, the highest hysteresis level should be used to avoid glitches on the output.

BIASPROG BIASPROG	Bias Current (μA), HYSTSEL=0			
	FULLBIAS=0, HALFBIAS=1	FULLBIAS=0, HALFBIAS=0	FULLBIAS=1, HALFBIAS=1	FULLBIAS=1, HALFBIAS=0
0b0000	0.05	0.1	3.3	6.5
0b0001	0.1	0.2	6.5	13
0b0010	0.2	0.4	13	26
0b0011	0.3	0.6	20	39
0b0100	0.4	0.8	26	52
0b0101	0.5	1.0	33	65
0b0110	0.6	1.2	39	78
0b0111	0.7	1.4	46	91
0b1000	1.0	2.0	65	130
0b1001	1.1	2.2	72	143
0b1010	1.2	2.4	78	156
0b1011	1.3	2.6	85	169
0b1100	1.4	2.8	91	182
0b1101	1.5	3.0	98	195
0b1110	1.6	3.2	104	208
0b1111	1.7	3.4	111	221

Figure 267: Bias Configuration

Z.3.3 Hysteresis

In the analog comparator, hysteresis can be configured to 8 different levels, including off which is level 0, through the HYSTSEL field in ACMPn_CTRL. When the hysteresis level is set above 0, 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 268). This feature can be used to filter out uninteresting input fluctuations around zero and only show changes that are big enough to breach the hysteresis threshold. Note that the ACMP current consumption will be influenced by the selected hysteresis level and in general decrease with increasing HYSTSEL values.

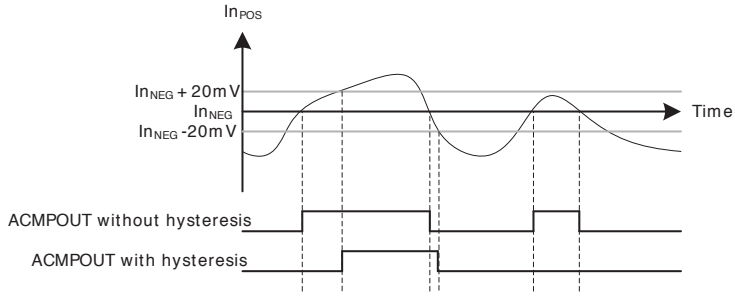


Figure 268: 20 mV Hysteresis Selected

Z.3.4 Input Selection

The POSSEL and NEGSEL fields in ACMPn_INPUTSEL controls which signals are connected to the two inputs of the comparator. 8 external pins are available for both the negative and positive input. For the negative input, 5 additional internal reference sources are available; 1.25 V bandgap, 2.5V bandgap, DAC channel 0, DAC channel 1, and V_{DD}. The V_{DD} reference can be scaled by a configurable factor, which is set in VDDLEVEL (in ACMPn_INPUTSEL) according to the following formula:

$$V_{\text{DD\_SCALED}} = V_{\text{DD}} \times VDDLEVEL/63 \quad (37)$$

A low power reference mode can be enabled by setting the LPREF bit in ACMPn_INPUTSEL. In this mode, the power consumption in the reference buffer (V_{DD} and bandgap) is lowered at the cost of accuracy. Low power mode will only save power if V_{DD} with VDDLEVEL higher than 0 or a bandgap reference is selected.

Normally the analog comparator input mux is disabled when the EN (in ACMPn_CTRL) bit is set low. However if the MUXEN bit in ACMPn_CTRL is set, the mux is enabled regardless of the EN bit. This will minimize kickback noise on the mux inputs when the EN bit is toggled.

Z.3.5 Capacitive Sense Mode

The analog comparator includes specialized hardware for capacitive sensing of passive push buttons. Such buttons are traces on 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 269). 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 (e.g. through PRS), the change in capacitance can be calculated.

The analog comparator contains a complete 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 one of four values by configuring the CSRESSEL bits in ACMPn_INPUTSEL. If the internal resistor is not enabled, the circuit will be open. The capacitive sense mode is enabled by setting the NEGSEL field in ACMPn_INPUTSEL to CAPSENSE. The input pin is selected through the POSSEL bits in ACMPn_INPUTSEL. The scaled V_{DD} in Figure 269 can be altered by configuring the VDDLEVEL in ACMPn_INPUTSEL. It is recommended to set the hysteresis (HYSTSEL in ACMPn_CTRL) higher than the lowest level when using the analog comparator in capacitive sense mode.

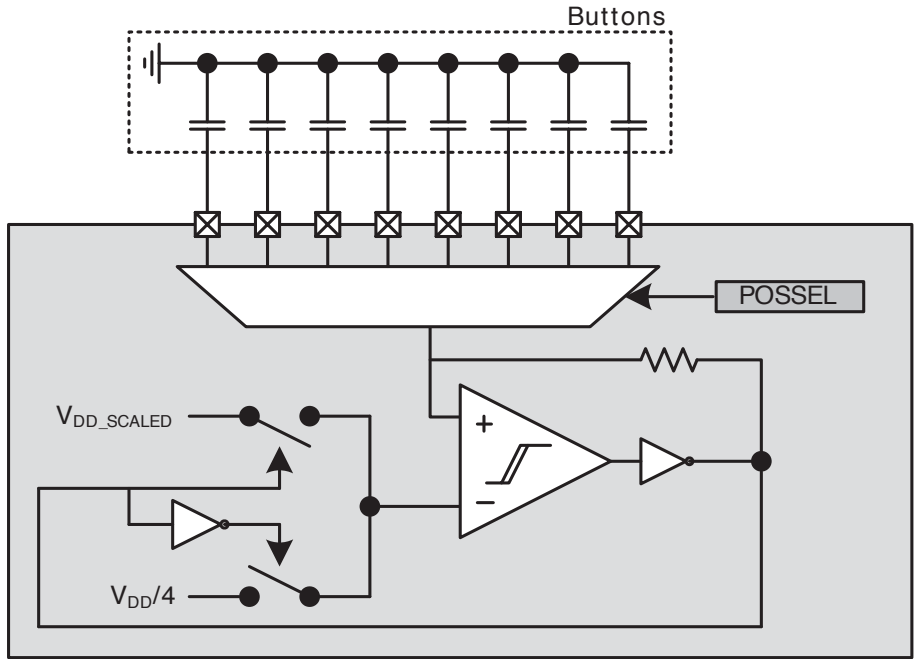


Figure 269: Capacitive Sensing Set-up

Z.3.6 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-EM1.

The analog comparator also includes an 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 comparator output is also available as a PRS signal.

Z.3.7 Output to GPIO

The output from the comparator is available as alternate function to the GPIO pins. Set the ACMPPEN bit in ACMPn_ROUTE to enable output to pin, and the LOCATION bits to select 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.

Z.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_IEN	RW	Interrupt Enable Register
0x010	ACMPn_IF	R	Interrupt Flag Register
0x014	ACMPn_IFS	W1	Interrupt Flag Set Register
0x018	ACMPn_IFC	W1	Interrupt Flag Clear Register
0x01C	ACMPn_ROUTE	RW	I/O Routing Register

Z.5 Register Description

Z.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

Offset	Bit Position															
Reset	0	1		0x7		0	0		0x0		0x0		0	0	0	0
Access	RW	RW		RW		RW	RW		RW		RW		RW	RW	RW	RW
Name	FULLBIAS	HALFBIAS		BIASPROG		IFALL	IRISE		WARMTIME		HYSTSEL		GPIOINV	INACTVAL	MODEEN	EN

Bit	Name	Reset	Access	Description
31	FULLBIAS	0	RW	**Full Bias Current**

Set this bit to 1 for full bias current in accordance with Table 267.

Bit	Name	Reset	Access	Description
30	HALFBIAS	1	RW	**Half Bias Current**

Set this bit to 1 to halve the bias current in accordance with Table 267.

Bit	Name	Reset	Access	Description
29:28	Reserved	To ensure compatibility with future devices, always write bits to 0.		
27:24	BIASPROG	0x7	RW	**Bias Configuration**

These bits control the bias current level in accordance with Table 267.

Bit	Name	Reset	Access	Description
23:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17	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.

Bit	Name	Reset	Access	Description
16	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.

Bit	Name	Reset	Access	Description
15:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	WARMTIME	0x0	RW	**Warm-up Time**

Set analog comparator warm-up time.

Value	Mode	Description
0	4CYCLES	4 HFPERCLK cycles.
1	8CYCLES	8 HFPERCLK cycles.
2	16CYCLES	16 HFPERCLK cycles.
3	32CYCLES	32 HFPERCLK cycles.
4	64CYCLES	64 HFPERCLK cycles.
5	128CYCLES	128 HFPERCLK cycles.
6	256CYCLES	256 HFPERCLK cycles.
7	512CYCLES	512 HFPERCLK cycles.

Bit	Name	Reset	Access	Description
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:4	HYSTSEL	0x0	RW	**Hysteresis Select**

Select hysteresis level. 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	~15 mV hysteresis.
2	HYST2	~22 mV hysteresis.
3	HYST3	~29 mV hysteresis.
4	HYST4	~36 mV hysteresis.
5	HYST5	~43 mV hysteresis.
6	HYST6	~50 mV hysteresis.
7	HYST7	~57 mV hysteresis.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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	MUXEN	0	RW	**Input Mux Enable**
	Enable Input Mux. Setting the EN bit will also enable the input mux.			
0	EN	0	RW	**Analog Comparator Enable**
	Enable/disable analog comparator.			

Z.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								1			0x00						0x8					0x0		
Access			RW					RW								RW			RW						RW					RW		
Name			CSRESSEL					CSRESEN								LPREF			VDDLEVEL						NEGSEL					POSSEL		

Bit	Name	Reset	Access	Description
31:30	Reserved	To ensure compatibility with future devices, always write bits to 0.		
29:28	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 datasheets.			

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.

27:25	Reserved	To ensure compatibility with future devices, always write bits to 0.		
24	CSRESEN	0	RW	**Capacitive Sense Mode Internal Resistor Enable**
	Enable/disable the internal capacitive sense resistor.			
23:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	LPREF	1	RW	**Low Power Reference Mode**
	Enable low power mode for VDD and bandgap references.			

Value	Description
0	Low power mode disabled.
1	Low power mode enabled.

15:14	Reserved	To ensure compatibility with future devices, always write bits to 0.		
13:8	VDDLEVEL	0x00	RW	**VDD Reference Level**
	Select scaling factor for VDD reference level.$V_{DD\_SCALED} = V_{DD} \times VDDLEVEL/63$.			
7:4	NEGSEL	0x8	RW	**Negative Input Select**
	Select negative input.			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	CH0	Channel 0 as negative input.
1	CH1	Channel 1 as negative input.
2	CH2	Channel 2 as negative input.
3	CH3	Channel 3 as negative input.
4	CH4	Channel 4 as negative input.
5	CH5	Channel 5 as negative input.
6	CH6	Channel 6 as negative input.
7	CH7	Channel 7 as negative input.
8	1V25	1.25 V as negative input.
9	2V5	2.5 V as negative input.
10	VDD	Scaled VDD as negative input.
11	CAPSENSE	Capacitive sense mode.
12	DAC0CH0	DAC0 channel 0.
13	DAC0CH1	DAC0 channel 1.

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	POSSEL	0x0	RW	**Positive Input Select**

Select positive input.

Value	Mode	Description
0	CH0	Channel 0 as positive input.
1	CH1	Channel 1 as positive input.
2	CH2	Channel 2 as positive input.
3	CH3	Channel 3 as positive input.
4	CH4	Channel 4 as positive input.
5	CH5	Channel 5 as positive input.
6	CH6	Channel 6 as positive input.
7	CH7	Channel 7 as positive input.

Z.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
Access																															R	R
Name																															ACMPOUT	ACMPACT

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	ACMPOUT	0	R	**Analog Comparator Output**
	Analog comparator output value.			
0	ACMPACT	0	R	**Analog Comparator Active**
	Analog comparator active status.			

Z.5.4 ACMPn_IEN - Interrupt Enable 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

Offset	Bit Position		
Access		RW	RW
Name		WARMUP	EDGE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	WARMUP	0	RW	**Warm-up Interrupt Enable**
	Enable/disable interrupt on finished warm-up.			
0	EDGE	0	RW	**Edge Trigger Interrupt Enable**
	Enable/disable edge triggered interrupt.			

Z.5.5 ACMPn_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	0
Access																															R	R
Name																															WARMUP	EDGE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			

Z.5.6 ACMPn_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	0
Access																															W1	W1
Name																															WARMUP	EDGE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	WARMUP	0	W1	**Warm-up Interrupt Flag Set**
	Write to 1 to set warm-up finished interrupt flag.			
0	EDGE	0	W1	**Edge Triggered Interrupt Flag Set**
	Write to 1 to set edge triggered interrupt flag.			

Z.5.7 ACMPn_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	0
Access																															W1	W1
Name																															WARMUP	EDGE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	WARMUP	0	W1	**Warm-up Interrupt Flag Clear**
	Write to 1 to clear warm-up finished interrupt flag.			
0	EDGE	0	W1	**Edge Triggered Interrupt Flag Clear**
	Write to 1 to clear edge triggered interrupt flag.			

Z.5.8 ACMPn_ROUTE - I/O Routing 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										0
Access																						RW										RW
Name																						LOCATION										ACMPPEN

Bit	Name	Reset	Access	Description
31:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	LOCATION	0x0	RW	**I/O Location**
	Decides the location of the ACMP I/O pin.			

Value	Mode	Description
0	LOC0	Location 0
1	LOC1	Location 1
2	LOC2	Location 2

Bit	Name	Reset	Access	Description
7:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	ACMPPEN	0	RW	**ACMP Output Pin Enable**
	Enable/disable analog comparator output to pin.			

AA ARM Voltage Comparator

AA.1 Introduction

The Voltage Supply Comparator is used to monitor the supply voltage from software. An interrupt can be generated when the supply falls below or rises above a programmable threshold.

VCMP comes in addition to the Power-on Reset and Brown-out Detector peripherals, that both generate reset signals when the voltage supply is insufficient for reliable operation. VCMP does not generate reset, only interrupt. Also note that the ADC is capable of sampling the input voltage supply.

AA.2 Features

- Voltage supply monitoring
- Scalable V_{DD} in 64 steps selectable as positive comparator input
- Internal 1.25 V bandgap reference
- Low power mode for internal V_{DD} and bandgap references
- Selectable hysteresis
  - 0 or $\pm$20 mV
- Selectable response time
- Asynchronous interrupt generation on selectable edges
  - Rising edge
  - Falling edge
  - Rising and Falling edges
- Operational in EM0-EM3
- Comparator output direct on PRS
- Configurable output when inactive to avoid unwanted interrupts

AA.3 Functional Description

An overview of the VCMP is shown in Figure 270.

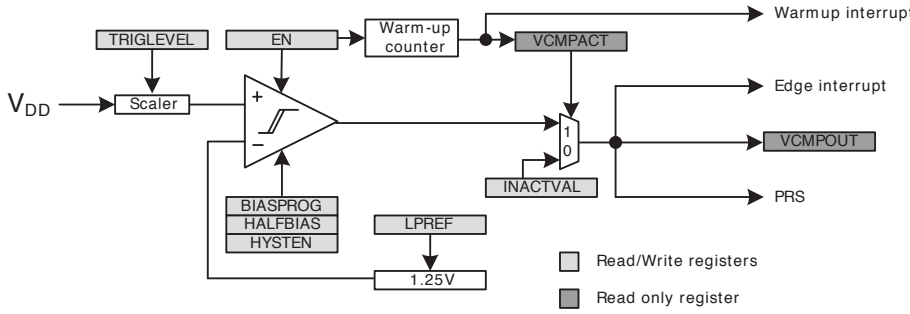


Figure 270: VCMP Overview

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 negative input voltage, the digital output is high and vice versa.

The output of the comparator can be read in the VCMPOUT bit in VCMP_STATUS. Configuration registers should only be changed while the comparator is disabled.

AA.3.1 Warm-up Time

VCMP is enabled by setting the EN bit in VCMP_CTRL. When this bit is set, the comparator must stabilize before becoming active and the outputs can be used. This time period is called the warm-up time. The warm-up time is a configurable number of HFPERCLK cycles, set in WARMTIME, which should be set to at least 10 μs. When the comparator is enabled and warmed up, the VCMPACT bit in VCMP_STATUS will be set to indicate that the comparator is active.

As long as the comparator is not enabled or not warmed up, VCMPACT will be cleared and the comparator output value is set to the value in INACTVAL in VCMP_CTRL.

One 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.

AA.3.2 Response Time

There is a delay from when the actual input voltage changes polarity, to when the output toggles. This period is called the response time and can be altered by increasing or decreasing the bias current to the comparator through the BIAS and HALFBIAS fields in VCMP_CTRL as shown in Table 271. Setting a lower bias current will result in lower power consumption, but a longer response time.

BIAS	Bias Current (μA)	
	HALFBIAS=0	HALFBIAS=1
0b0000	0.1	0.05
0b0001	0.2	0.1
0b0010	0.4	0.2
0b0011	0.6	0.3
0b0100	0.8	0.4
0b0101	1.0	0.5
0b0110	1.2	0.6
0b0111	1.4	0.7
0b1000	2.0	1.0
0b1001	2.2	1.1
0b1010	2.4	1.2
0b1011	2.6	1.3
0b1100	2.8	1.4
0b1101	3.0	1.5
0b1110	3.2	1.6
0b1111	3.4	1.7

Figure 271: Bias Configuration

AA.3.3 Hysteresis

In the voltage supply comparator, hysteresis can be enabled by setting HYSTEN in VCMP_CTRL. When HYSTEN is set, the digital output will not toggle until the positive input voltage is at least 20mV above or below the negative input voltage. This feature can be used to filter out uninteresting input fluctuations around zero and only show changes that are big enough to breach the hysteresis threshold.

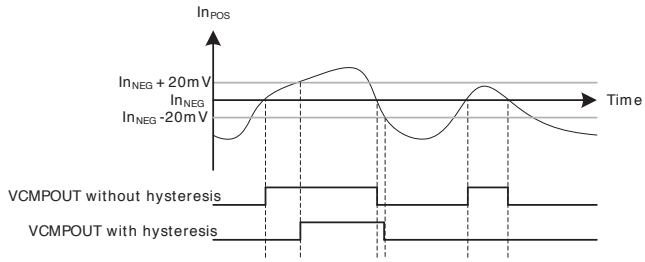


Figure 272: VCMP 20 mV Hysteresis Enabled

AA.3.4 Input Selection

The positive comparator input is always connected to the scaled power supply input. The negative comparator input is connected to the internal 1.25 V bandgap reference. The V_{DD} trigger level can be configured by setting the TRIGLEVEL field in VCMP_CTRL according to the following formula:

$$V_{\text{DD Trigger Level}} = 1.667V + 0.034V \times TRIGLEVEL \quad (38)$$

A low power reference mode can be enabled by setting the LPREF bit in VCMP_INPUTSEL. In this mode, the power consumption in the reference buffer (V_{DD} and bandgap) is lowered at the cost of accuracy.

AA.3.5 Interrupts and PRS Output

The VCMP includes an edge triggered interrupt flag (EDGE in VCMP_IF). If either IRISE and/or IFALL in VCMPn_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 VCMP_IF is set and enabled through the EDGE bit in VCMPn_IEN. The edge interrupt can also be used to wake up the device from EM3-EM1. VCMP also includes an interrupt flag, WARMUP in VCMP_IF, which is set when a warm-up sequence has finished. An interrupt request will be sent if the WARMUP interrupt flag in VCMP_IF is set and enabled through the WARMUP bit in VCMPn_IEN. The synchronized comparator output is also available as a PRS output signal.

AA.4 Register Map

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

Offset	Name	Type	Description
0x000	VCMP_CTRL	RW	Control Register
0x004	VCMP_INPUTSEL	RW	Input Selection Register
0x008	VCMP_STATUS	R	Status Register
0x00C	VCMP_IEN	RW	Interrupt Enable Register
0x010	VCMP_IF	R	Interrupt Flag Register
0x014	VCMP_IFS	W1	Interrupt Flag Set Register
0x018	VCMP_IFC	W1	Interrupt Flag Clear Register

AA.5 Register Description

AA.5.1 VCMP_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		1			0x7										0	0						0x0						0		0		0
Access		RW			RW										RW	RW						RW						RW		RW		RW
Name		HALFBIAS			BIASPROG										IFALL	IRISE						WARMTIME						HYSTEN		INACTVAL		EN

Bit	Name	Reset	Access	Description
31	Reserved	To ensure compatibility with future devices, always write bits to 0.		
30	HALFBIAS	1	RW	**Half Bias Current**
	Set this bit to 1 to halve the bias current. Table 271.			
29:28	Reserved	To ensure compatibility with future devices, always write bits to 0.		
27:24	BIASPROG	0x7	RW	**VCMP Bias Programming Value**
	These bits control the bias current level. Table 271.			
23:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17	IFALL	0	RW	**Falling Edge Interrupt Sense**
	Set this bit to 1 to set the EDGE interrupt flag on falling edges of comparator output.			
16	IRISE	0	RW	**Rising Edge Interrupt Sense**
	Set this bit to 1 to set the EDGE interrupt flag on rising edges of comparator output.			
15:11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	WARMTIME	0x0	RW	**Warm-Up Time**
	Set warm-up time			

Value	Mode	Description
0	4CYCLES	4 HFPERCLK cycles
1	8CYCLES	8 HFPERCLK cycles
2	16CYCLES	16 HFPERCLK cycles
3	32CYCLES	32 HFPERCLK cycles
4	64CYCLES	64 HFPERCLK cycles
5	128CYCLES	128 HFPERCLK cycles
6	256CYCLES	256 HFPERCLK cycles
7	512CYCLES	512 HFPERCLK cycles

Bit	Name	Reset	Access	Description
7:5	Reserved	To ensure compatibility with future devices, always write bits to 0.		
4	HYSTEN	0	RW	**Hysteresis Enable**
	Enable hysteresis.			

Value	Description
0	No hysteresis
1	+-20 mV hysteresis

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	INACTVAL	0	RW	**Inactive Value**
	Configure the output value when the comparator is inactive.			
1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	EN	0	RW	**Voltage Supply Comparator Enable**
	Enable/disable voltage supply comparator.			

AA.5.2 VCMP_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																								0			0x00					
Access																								RW			RW					

Offset	Bit Position																													
Name																								LPREF					TRIGLEVEL	

Bit	Name	Reset	Access	Description
31:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8	LPREF	0	RW	**Low Power Reference**
	Enable/disable low power mode for VDD and bandgap reference. When using this bit, always leave it as 0 during warm-up and then set it to 1 if desired when the warm-up is done.			
7:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:0	TRIGLEVEL	0x00	RW	**Trigger Level**
	Select VDD trigger level. V_{trig} = 1.667V+0.034V×TRIGLEVEL.			

AA.5.3 VCMP_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																															VCMPOUT	VCMPACT

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	VCMPOUT	0	R	**Voltage Supply Comparator Output**
	Voltage supply comparator output value			
0	VCMPACT	0	R	**Voltage Supply Comparator Active**
	Voltage supply comparator active status.			

AA.5.4 VCMP_IEN - Interrupt Enable 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
Access																															RW	RW
Name																															WARMUP	EDGE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	WARMUP	0	RW	**Warm-up Interrupt Enable**
	Enable/disable interrupt on finished warm-up.			
0	EDGE	0	RW	**Edge Trigger Interrupt Enable**
	Enable/disable edge triggered interrupt.			

AA.5.5 VCMP_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	0
Access																															R	R
Name																															WARMUP	EDGE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	WARMUP	0	R	**Warm-up Interrupt Flag**
	Indicates that warm-up has finished.			
0	EDGE	0	R	**Edge Triggered Interrupt Flag**
	Indicates that there has been a rising and/or falling edge on the VCMP output.			

AA.5.6 VCMP_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	0
Access																															W1	W1
Name																															WARMUP	EDGE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	WARMUP	0	W1	**Warm-up Interrupt Flag Set**
	Write to 1 to set warm-up finished interrupt flag			
0	EDGE	0	W1	**Edge Triggered Interrupt Flag Set**
	Write to 1 to set edge triggered interrupt flag			

AA.5.7 VCMP_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	0
Access																															W1	W1
Name																															WARMUP	EDGE

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	WARMUP	0	W1	**Warm-up Interrupt Flag Clear**
	Write to 1 to clear warm-up finished interrupt flag			

Bit	Name	Reset	Access	Description
0	EDGE	0	W1	**Edge Triggered Interrupt Flag Clear**
	Write to 1 to clear edge triggered interrupt flag			

AB ARM Analog to Digital Converter

AB.1 Introduction

The ADC is a Successive Approximation Register (SAR) architecture, with a resolution of up to 12 bits at up to one million samples per second. The integrated input mux can select inputs from 8 external pins and 6 internal signals.

AB.2 Features

- Programmable resolution (6/8/12-bit)
  - 13 prescaled clock (ADC_CLK) cycles per conversion
  - Maximum 1 MSPS @ 12-bit
  - Maximum 1.86 MSPS @ 6-bit
- Configurable acquisition time
- Integrated prescaler
  - Selectable clock division factor from 1 to 128
- 13 MHz to 32 kHz allowed for ADC_CLK
- 18 input channels
  - 8 external channels
    - single ended or 4 differential channels
  - 6 internal single ended channels
    - Including temperature sensor
- Integrated input filter
  - Low pass RC filter
  - Decoupling capacitor
- Left or right adjusted results
  - Results in 2's complement representation
  - Differential results sign extended to 32-bit results
- Programmable scan sequence
  - Up to 8 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
  - Overflow interrupt flag set when overwriting unread results
  - Conversion tailgating support for predictable periodic scans

- Programmable single conversion
  - Triggered by software or PRS input
  - Can be interleaved between two scan sequences
  - One shot or repetitive mode
  - Oversampling available
  - Overflow interrupt flag set when overwriting unread results
- 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 sample mode
  - Common OSR select
- Individually selectable voltage reference for scan and single mode
  - Internal 1.25V reference
  - Internal 2.5V reference
  - V_{DD}
  - Internal 5 V differential reference
  - Single ended external reference
  - Differential external reference
  - Unbuffered $2xV_{DD}$
- Support for offset and gain calibration
- Interrupt generation and/or DMA request
  - Finished single conversion
  - Finished scan conversion
  - Single conversion results overflow
  - Scan sequence results overflow
- Loopback configuration with DAC output measurement

AB.3 Functional Description

An overview of the ADC is shown in Figure 273.

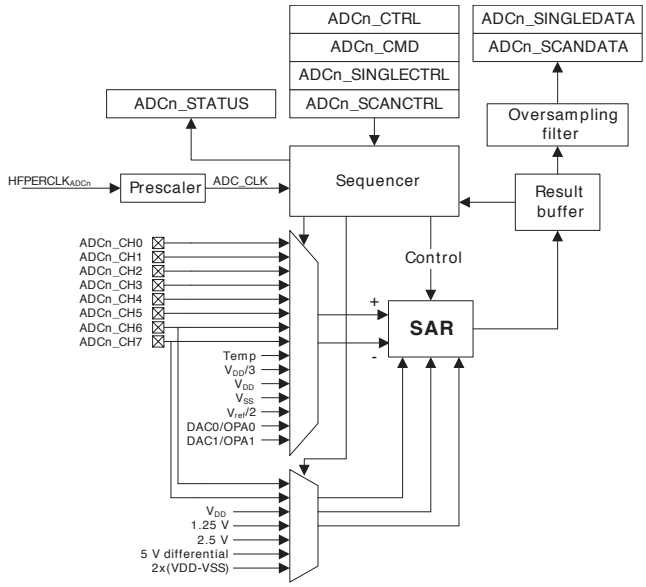


Figure 273: ADC Overview

AB.3.1 Clock Selection

The ADC has an internal prescaler (PRESC bits in ADCn_CTRL) which can divide the peripheral clock (HFPERCLK) by any factor between 1 and 128. Note that the resulting ADC_CLK should not be set to a higher frequency than 13 MHz and not lower than 32 kHz.

AB.3.2 Conversions

A conversion consists of two phases. 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 and single conversions (see Section AB.3.7) by setting AT in ADCn_SINGLECTRL/ADCn_SCANCTRL. The acquisition times can be set to any integer power of 2 from 1 to 256 ADC_CLK cycles.

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 the internal temperature sensor and $V_{dd}/3$ is given in the electrical characteristics for the device.

The analog to digital converter core uses one clock cycle per output bit in the approximation phase.

$$T_{\text{conv}} = (T_{\text{A}} + N)xOSR \quad (39)$$

T_A equals the number of acquisition cycles and N is the resolution. OSR is the oversampling ratio (see Section AB.3.7). The minimum conversion time is 7 ADC_CYCLES with 6 bit resolution and 13 ADC_CYCLES with 12 bit resolution. The maximum conversion time is 1097728 ADC_CYCLES with the longest acquisition time, 12 bit resolution and highest oversampling rate.

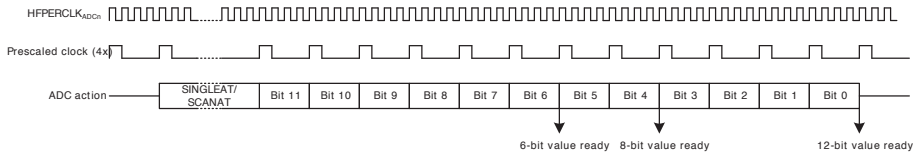


Figure 274: ADC Conversion Timing

AB.3.3 Warm-up Time

The ADC needs to be warmed up some time before a conversion can take place. This time period is called the warm-up time. When enabling the ADC or changing references between samples, the ADC is automatically warmed up for 1μs and an additional 5 μs if the bandgap is selected as reference.

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, eliminating the need for warm-up. Figure 275 shows the analog power consumption in scenarios using the different WARMUPMODE settings.

Only the bandgap reference selected for scan mode can be kept warm. If a different bandgap reference is selected for single mode, the warm-up time still applies.

- NORMAL: ADC and references are shut off when there are no samples waiting. a) in Figure 275 shows this mode used with an internal bandgap reference. Figure d) shows this mode when using VDD or an external reference.
- FASTBG: Bandgap warm-up is eliminated, but with reduced reference accuracy. d) in Figure 275 shows this mode used with an internal bandgap reference.
- KEEPSCANREFWARM: The reference selected for scan mode is kept warm. The ADC will still need to be warmed up before conversion. b) in Figure 275 shows this mode used with an internal bandgap reference.
- KEEPADCWARM: The ADC and the reference selected for scan mode is kept warm. c) in Figure 275 shows this mode used with an internal bandgap reference.

The minimum warm-up times are given in μs. The timing is done automatically by the ADC, given that a proper time base is given in the TIMEBASE bits in ADCn_CTRL. The TIMEBASE must be set to the number of HFPERCLK which corresponds to at least 1 μs. The TIMEBASE only affects the timing of the warm-up sequence and not the ADC_CLK.

When entering Energy Modes 2 or 3, the ADC must be stopped and WARMUPMODE in ADCn_CTRL written to 0.

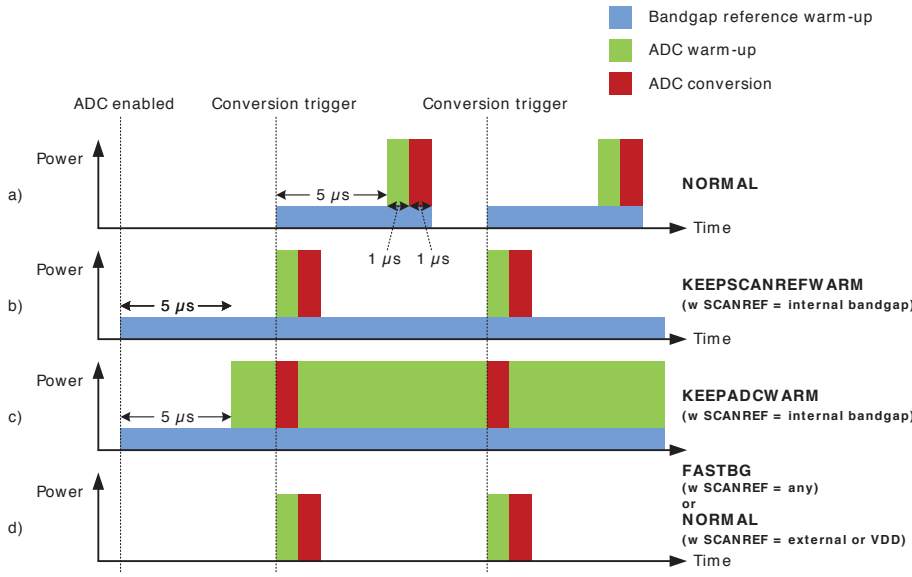


Figure 275: ADC Analog Power Consumption With Different WARMUP-MODE Settings

AB.3.4 Input Selection

The ADC is connected to 8 external input pins, which can be selected as 8 different single ended inputs or 4 differential inputs. In addition, 6 single ended internal inputs can be selected. The available selections are given in the register description for ADCn_SINGLECTRL and ADCn_SCANCTRL.

For offset calibration purposes it is possible to internally short the differential ADC inputs and thereby measure a 0 V differential. Differential 0 V is selected by writing the DIFF bit to 1 and INPUTSEL to 4 in ADCn_SINGLECTRL. Calibration is described in detail in Section AB.3.10.

When VDD/3 is sampled, the acquisition time should be above a lower limit. The reader is referred to the datasheet for minimum VDD/3 acquisition time.

Input Filtering

The selected input signal can be filtered, either through an internal low pass RC filter or an internal decoupling capacitor. The different filter configurations can be enabled through the LPFMODE bits in ADCn_CTRL. For maximum SNR, LPFMODE is recommended set to DECAP, with a cutoff frequency of 31.5 MHz.

The RC input filter configuration is given in Figure 276. The resistance and capacitance values are given in the electrical characteristics for the device, named $R_{ADCFILT}$ and $C_{ADCFILT}$ respectively.

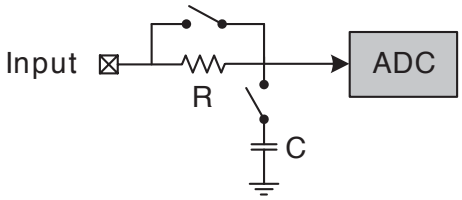


Figure 276: ADC RC Input Filter Configuration

Temperature Measurement

The ADC includes an internal temperature sensor. This sensor is characterized during production and the temperature readout from the ADC at production temperature, ADC0_TEMP_0_READ_1V25, is given in the Device Information (DI) page. The production temperature, CAL_TEMP_0, is also given in this page. The temperature gradient, TGRAD_ADCTH (mV/degree Celsius), for the sensor is found in the datasheet for the devices. By selecting 1.25 V internal reference and measuring the internal temperature sensor with 12 bit resolution, the temperature can be calculated according to the following formula:

$$T_{CELSIUS} = CAL\_TEMP\_0 - (ADC0\_TEMP\_0\_READ\_1V25 - ADC\_result) \times Vref/(4096 \times$$

The minimum acquisition time for the temperature reference is found in the electrical characteristics for the device.

AB.3.5 Reference Selection

The reference voltage can be selected from these sources:

- 1.25 V internal bandgap.
- 2.5 V internal bandgap.
- V_{DD}.
- 5 V internal differential bandgap.
- External single ended input from Ch. 6.
- Differential input, 2x(Ch. 6 - Ch. 7).
- Unbuffered $2xV_{DD}$.
- The 2.5 V reference needs a supply voltage higher than 2.5 V.
- The differential 5 V reference needs a supply voltage higher than 2.75 V.

Since the $2xV_{DD}$ differential reference is unbuffered, it is directly connected to the ADC supply voltage and more susceptible to supply noise. The V_{DD} reference is buffered both in single ended and differential mode.

If a differential reference with a larger range than the supply voltage is combined with single ended measurements, for instance the 5 V internal reference, the full ADC range will not be available because the maximum input voltage is limited by the maximum electrical ratings.

Single ended measurements with the external differential reference are not supported.

AB.3.6 Programming of Bias Current

The bias current of the bandgap reference and the ADC comparator can be scaled by the BIASPROG, HALFBIAS and COMPBIAS bit fields of the ADCn_BIASPROG register. The BIASPROG and HALFBIAS bitfields scale the current of ADC bandgap reference, and the COMPBIAS bits provide an additional bias programming for the ADC comparator as illustrated in Figure 277. The electrical characteristics given in the datasheet require the bias configuration to be set to the default values, where no other bias values are given.

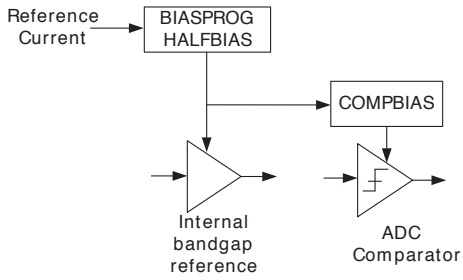


Figure 277: ADC Bias Programming

The minimum value of the BIASPROG and COMPBIAS bitfields of the ADCn_BIASPROG register (i.e. BIASPROG=0b0000, COMPBIAS=0b0000) represent the minimum bias currents. Similarly BIASPROG=0b1111 and COMPBIAS=0b1111 represent the maximum bias currents. Additionally, the bias current defined by the BIASPROG setting can be halved by setting the HALFBIAS bit of the ADCn_BIASPROG register.

The bias current settings should only be changed while the ADC is disabled.

AB.3.7 ADC Modes

The ADC contains two separate programmable modes, one single sample mode and one scan mode. Both modes have separate configuration and result registers and can be set up to run only once per trigger or repetitively. The scan mode has priority over the single sample mode. However, if scan sequence is running, a triggered single sample will be interleaved between two scan samples.

Single Sample Mode

The single sample mode can be used to convert a single sample either once per trigger or repetitively. The configuration of the single sample mode is done in the ADCn_SINGLECTRL register and the results are found in the ADCn_SINGLEDATA register. The SINGLEDV bit in ADCn_STATUS is set high when there is valid data in the result register and is cleared when the data is read. The single mode results can also be read through ADCn_SINGLEDATAP without SINGLEDV being cleared. DIFF in ADCn_SINGLECTRL selects whether differential or single ended inputs are used and INPUTSEL selects input pin(s).

Scan mode

The scan mode is used to perform sweeps of the inputs. The configuration of the scan sequence is done in the ADCn_SCANCTRL register and the results are found in the ADCn_SCANDATA register. The SCANDV bit in ADCn_STATUS is set high when there is valid data in the result register and is cleared when the data is read. The scan mode results can also be read through ADCn_SCANDATAP without SCANDV being cleared. The inputs included in the sequence are defined by a the mask in INPUTMASK in ADCn_SCANCTRL. When the scan sequence is triggered, the sequence samples all inputs that are included in the mask, starting at the lowest pin number. DIFF in ADCn_SCANCTRL selects whether single ended or differential inputs are used.

Conversion Tailgating

The scan sequence has priority over the single sample mode. However, a scan trigger will not interrupt in the middle of a single conversion. If a scan sequence is triggered by a timer on a periodic basis, single sample 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 samples will wait for the next scan sequence to finish before activating (see Figure 278). The single sample will then follow immediately after the scan sequence. In this way, the scan sequence will always start immediately when triggered, if the period between the scan triggers is big enough to allow any single samples that might be triggered to finish in between the scan sequences.

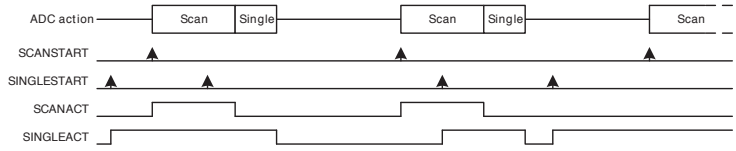


Figure 278: ADC Conversion Tailgating

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 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 system requires one HFPERCLK cycle pulses to trigger conversions. Setting PRSEN in ADCn_SINGLECTRL/ADCn_SCANC enables triggering from PRS input. Which PRS channel to listen to is defined by PRSSEL in ADCn_SINGLECTRL/ADCn_SCANCTRL. When PRS trigger is selected, it is still possible to trigger the conversion from software. The reader is referred to the PRS datasheet for more information on how to set up the PRS channels.

The conversion settings should not be changed while the ADC is running as this can lead to unpredictable behavior.

The prescaled clock phase is always reset by a triggered conversion as long as a conversion is not ongoing. This gives predictable latency from the time of the trigger to the time the conversion starts, regardless of when in the prescaled clock cycle the trigger occur.

Results

The results are presented in 2's complement form and the format for differential and single ended mode is given in Table 279 and Table 280. If differential mode is selected, the results are sign extended up to 32-bit (shown in Table 282).

Figure 279: ADC Single Ended Conversion

Input/Reference	Results	
	Binary	Hex value
1	111111111111	FFF
0.5	011111111111	7FF
1/4096	000000000001	001
0	000000000000	000

Figure 280: ADC Differential Conversion

Input/Reference	Results	
	Binary	Hex value
0.5	011111111111	7FF
0.25	001111111111	3FF
1/2048	000000000001	001
0	000000000000	000
-1/2048	111111111111	FFF
-0.25	101111111111	BFF
-0.5	100000000000	800

Resolution

The ADC gives out 12-bit results, by default. However, if full 12-bit resolution is not needed, it is possible to speed up the conversion by selecting a lower resolution

(N = 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.

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 sample mode (OVSRSEL field in ADCn_CTRL).

With oversampling, each selected input is sampled a number (given by the OVSR) of times, 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 281.

Figure 281: 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

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 282. 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).

Adjustment	Resolution	31	30	29	28	27	26	25	24	23	22
Right	12	11	11	11	11	11	11	11	11	11	11
	8	7	7	7	7	7	7	7	7	7	7
	6	5	5	5	5	5	5	5	5	5	5
	OVS	15	15	15	15	15	15	15	15	15	15
Left	12	11	11	11	11	11	11	11	11	11	11
	8	7	7	7	7	7	7	7	7	7	7
	6	5	5	5	5	5	5	5	5	5	5
	OVS	15	15	15	15	15	15	15	15	15	15

Figure 282: ADC Results Representation

AB.3.8 Interrupts, PRS Output

The single and scan modes have separate interrupt flags indicating finished conversions. Setting one of these flags will result in an ADC interrupt if the corresponding interrupt enable bit is set in ADCn_IEN.

In addition to the finished conversion flags, there is a scan and single sample result overflow flag which signalizes that a result from a scan sequence or single sample has been overwritten before being read.

A finished conversion will result in a one HFPERCLK cycle pulse which is output to the Peripheral Reflex System (PRS).

AB.3.9 DMA Request

The ADC has two DMA request lines, SINGLE and SCAN, which are set when a single or scan conversion has completed. The request are cleared when the corresponding single or scan result register is read.

AB.3.10 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. The ADC calibration (ADCn_CAL) register contains four register fields for calibrating offset and gain for both single and scan mode. The gain and offset calibration are done in single mode, but the resulting calibration values can be used for both single and scan mode.

Gain and offset for the 1V25, 2V5 and VDD references 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.

The SCANGAIN and SINGLEGAIN calibration fields are not used when the unbuffered differential 2xVDD reference is selected.

The effects of changing the calibration register values are given in Table 283. Step by step calibration procedures for offset and gain are given in Section AB.3.10 and Section AB.3.10.

Figure 283: Calibration Register Effect

Calibration Register	ADC Result	Calibration Binary Value	Calibration Hex Value
[2Offset	Lowest Output	0111111	3F
[2-	Highest Output	1000000	40
[2Gain	Lowest Output	0000000	00
[2-	Highest Output	1111111	7F

The offset calibration register expects a signed 2's complement value with negative effect. A high value gives a low ADC reading.

The gain calibration register expects an unsigned value with positive effect. A high value gives a high ADC reading.

Offset Calibration

Offset calibration must be performed prior to gain calibration. Follow these steps for the offset calibration in single mode:

1. Select wanted reference by setting the REF bitfield of the ADCn_SINGLECTRL register.
2. Set the AT bitfield of the ADCn_SINGLECTRL register to 16CYCLES.
3. Set the INPUTSEL bitfield of the ADCn_SINGLECTRL register to DIFF0, and set the DIFF bitfield to 1 for enabling differential input. Since the input voltage is 0, the expected ADC output is the half of the ADC code range as it is in differential mode.
4. A binary search is used to find the offset calibration value. Set the SINGLESTART bit in the ADCn_CMD register and read the ADCn_SINGLEDATA register. The result of the binary search is written to the SINGLEOFFSET field of the ADCn_CAL register.

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 (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 range.
3. A binary search is used to find the gain calibration value. Set the SINGLESTART bit in the ADCn_CTRL register and read the ADCn_SINGLEDATA register. The target value is ideally the top of the ADC 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.

AB.4 Register Map

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

Offset	Name	Type	Description
0x000	ADCn_CTRL	RW	Control Register
0x004	ADCn_CMD	W1	Command Register
0x008	ADCn_STATUS	R	Status Register
0x00C	ADCn_SINGLECTRL	RW	Single Sample Control Register
0x010	ADCn_SCANCTRL	RW	Scan Control Register
0x014	ADCn_IEN	RW	Interrupt Enable Register
0x018	ADCn_IF	R	Interrupt Flag Register
0x01C	ADCn_IFS	W1	Interrupt Flag Set Register
0x020	ADCn_IFC	W1	Interrupt Flag Clear Register
0x024	ADCn_SINGLEDATA	R	Single Conversion Result Data
0x028	ADCn_SCANDATA	R	Scan Conversion Result Data
0x02C	ADCn_SINGLEDATAP	R	Single Conversion Result Data Peek Register
0x030	ADCn_SCANDATAP	R	Scan Sequence Result Data Peek Register
0x034	ADCn_CAL	RW	Calibration Register
0x03C	ADCn_BIASPROG	RW	Bias Programming Register

AB.5 Register Description

AB.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					0x1F								0x00									0x0		0		0x0	
Access					RW					RW								RW									RW		RW		RW	
Name					OVSRSEL					TIMEBASE								PRESC									LPFMODE		TAILGATE		WARMUPMODE	

Bit	Name	Reset	Access	Description

Bit	Name	Reset	Access	Description
31:28	Reserved	To ensure compatibility with future devices, always write bits to 0.		
27:24	OVSRSEL	0x0	RW	**Oversample Rate Select**

Select oversampling rate. Oversampling must be enabled for each mode for this setting to take effect.

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.		
22:16	TIMEBASE	0x1F	RW	**Time Base**

Set time base used for ADC warm up sequence according to the HFPERCLK frequency. The time base is defined as a number of HFPERCLK cycles which should be set equal to or higher than 1us.

Value	Description
TIMEBASE	ADC warm-up is set to TIMEBASE+1 HFPERCLK clock cycles and bandgap warm-up is set to 5x(TIMEBASE+1) HFPERCLK cycles.

15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14:8	PRESC	0x00	RW	**Prescaler Setting**

Select clock division factor.

Value	Description
PRESC	Clock division factor of PRESC+1.

7:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:4	LPFMODE	0x0	RW	**Low Pass Filter Mode**

These bits control the filtering of the ADC input. Details on the filter characteristics can be found in the device datasheets.

Value	Mode	Description
0	BYPASS	No filter or decoupling capacitor
1	DECAP	On chip decoupling capacitor selected
2	RCFILT	On chip RC filter selected

3	TAILGATE	0	RW	**Conversion Tailgating**

Enable/disable conversion tailgating.

Value	Description
0	Scan sequence has priority, but can be delayed by ongoing single samples.
1	Scan sequence has priority and single samples will only start immediately after scan sequence.

2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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
1	FASTBG	Bandgap references do not need warm up, but have reduced accuracy.
2	KEEPSCANREFWARM	Reference selected for scan mode is kept warm.
3	KEEPADCWARM	ADC is kept warmed up and scan reference is kept warm

AB.5.2 ADCn_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
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.		
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 Conversion Stop**
	Write a 1 to stop single conversion.			
0	SINGLESTART	0	W1	**Single Conversion Start**
	Write to 1 to start single conversion.			

AB.5.3 ADCn_STATUS - Status Register

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

Bit	Name	Reset	Access	Description
31:27	Reserved	To ensure compatibility with future devices, always write bits to 0.		
26:24	SCANDATASRC	0x0	R	**Scan Data Source**
	This value indicates from which input channel the results in the ADCn_SCANDATA register originates.			

Value	Mode	Description
0	CH0	Single ended mode: SCANDATA result originates from ADCn_CH0. Differential mode: SCANDATA result originates from ADCn_CH0-ADCn_CH1
1	CH1	Single ended mode: SCANDATA result originates from ADCn_CH1. Differential mode: SCANDATA result originates from ADCn_CH2_ADCn_CH3
2	CH2	Single ended mode: SCANDATA result originates from ADCn_CH2. Differential mode: SCANDATA result originates from ADCn_CH4-ADCn_CH5
3	CH3	Single ended mode: SCANDATA result originates from ADCn_CH3. Differential mode: SCANDATA result originates from ADCn_CH6-ADCn_CH7
4	CH4	SCANDATA result originates from ADCn_CH4
5	CH5	SCANDATA result originates from ADCn_CH5
6	CH6	SCANDATA result originates from ADCn_CH6
7	CH7	SCANDATA result originates from ADCn_CH7

23:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17	SCANDV	0	R	**Scan Data Valid**
	Scan conversion data is valid.			
16	SINGLEDV	0	R	**Single Sample Data Valid**

Bit	Name	Reset	Access	Description
	Single conversion data is valid.			
15:13	Reserved	To ensure compatibility with future devices, always write bits to 0.		
12	WARM	0	R	**ADC Warmed Up**
	ADC is warmed up.			
11:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9	SCANREFWARM	0	R	**Scan Reference Warmed Up**
	Reference selected for scan mode is warmed up.			
8	SINGLEREFWARM	0	R	**Single Reference Warmed Up**
	Reference selected for single mode is warmed up.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	SCANACT	0	R	**Scan Conversion Active**
	Scan sequence is active or has pending conversions.			
0	SINGLEACT	0	R	**Single Conversion Active**
	Single conversion is active or has pending conversions.			

AB.5.4 ADCn_SINGLECTRL - Single Sample 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					0x0							0x0						0x0			0	0	0
Access	RW							RW	RW					RW							RW						RW			RW	RW	RW
Name	PRSSEL							PRSEN	AT					REF							INPUTSEL						RES			ADJ	DIFF	REP

Bit	Name	Reset	Access	Description
31:28	PRSSEL	0x0	RW	**Single Sample PRS Trigger Select**
	Select PRS trigger for single sample.			

Value	Mode	Description
0	PRSCH0	PRS ch 0 triggers single sample
1	PRSCH1	PRS ch 1 triggers single sample
2	PRSCH2	PRS ch 2 triggers single sample
3	PRSCH3	PRS ch 3 triggers single sample
4	PRSCH4	PRS ch 4 triggers single sample
5	PRSCH5	PRS ch 5 triggers single sample
6	PRSCH6	PRS ch 6 triggers single sample
7	PRSCH7	PRS ch 7 triggers single sample
8	PRSCH8	PRS ch 8 triggers single sample
9	PRSCH9	PRS ch 9 triggers single sample
10	PRSCH10	PRS ch 10 triggers single sample
11	PRSCH11	PRS ch 11 triggers single sample

27:25	Reserved	To ensure compatibility with future devices, always write bits to 0.		
24	PRSEN	0	RW	**Single Sample PRS Trigger Enable**
	Enabled/disable PRS trigger of single sample.			

Value	Description
0	Single sample is not triggered by PRS input
1	Single sample is triggered by PRS input selected by PRSSEL

23:20	AT	0x0	RW	**Single Sample Acquisition Time**
	Select the acquisition time for single sample.			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	1CYCLE	1 ADC_CLK cycle acquisition time for single sample
1	2CYCLES	2 ADC_CLK cycles acquisition time for single sample
2	4CYCLES	4 ADC_CLK cycles acquisition time for single sample
3	8CYCLES	8 ADC_CLK cycles acquisition time for single sample
4	16CYCLES	16 ADC_CLK cycles acquisition time for single sample
5	32CYCLES	32 ADC_CLK cycles acquisition time for single sample
6	64CYCLES	64 ADC_CLK cycles acquisition time for single sample
7	128CYCLES	128 ADC_CLK cycles acquisition time for single sample
8	256CYCLES	256 ADC_CLK cycles acquisition time for single sample

Bit	Name	Reset	Access	Description
19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18:16	REF	0x0	RW	**Single Sample Reference Selection**

Select reference to ADC single sample mode.

Value	Mode	Description
0	1V25	Internal 1.25 V reference
1	2V5	Internal 2.5 V reference
2	VDD	Buffered VDD
3	5VDIFF	Internal differential 5 V reference
4	EXTSINGLE	Single ended external reference from pin 6
5	2XEXTDIFF	Differential external reference, 2x(pin 6 - pin 7)
6	2XVDD	Unbuffered 2xVDD

Bit	Name	Reset	Access	Description
15:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11:8	INPUTSEL	0x0	RW	**Single Sample Input Selection**

Select input to ADC single sample mode in either single ended mode or differential mode.

DIFF = 0		
Mode	Value	Description
CH0	0	ADCn_CH0
CH1	1	ADCn_CH1
CH2	2	ADCn_CH2
CH3	3	ADCn_CH3
CH4	4	ADCn_CH4
CH5	5	ADCn_CH5
CH6	6	ADCn_CH6
CH7	7	ADCn_CH7
TEMP	8	Temperature reference
VDDDIV3	9	VDD/3
VDD	10	VDD
VSS	11	VSS
VREFDIV2	12	VREF/2
DAC0OUT0	13	DAC0 output 0
DAC0OUT1	14	DAC0 output 1
DIFF = 1		
Mode	Value	Description
CH0CH1	0	Positive input: ADCn_CH0 Negative input: ADCn_CH1
CH2CH3	1	Positive input: ADCn_CH2 Negative input: ADCn_CH3
CH4CH5	2	Positive input: ADCn_CH4 Negative input: ADCn_CH5
CH6CH7	3	Positive input: ADCn_CH6 Negative input: ADCn_CH7
DIFF0	4	Differential 0 (Short between positive and negative inputs)

Bit	Name	Reset	Access	Description
7:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:4	RES	0x0	RW	**Single Sample Resolution Select**

Select single sample conversion resolution.

Bit	Name	Reset	Access	Description

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

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	ADJ	0	RW	**Single Sample Result Adjustment**

Select single sample result adjustment.

Value	Mode	Description
0	RIGHT	Results are right adjusted
1	LEFT	Results are left adjusted

Bit	Name	Reset	Access	Description
1	DIFF	0	RW	**Single Sample Differential Mode**

Select single ended or differential input.

Value	Description
0	Single ended input
1	Differential input

Bit	Name	Reset	Access	Description
0	REP	0	RW	**Single Sample Repetitive Mode**

Enable/disable repetitive single samples.

Value	Description
0	Single conversion mode is deactivated after one conversion
1	Single conversion mode is converting continuously until SINGLESTOP is written

AB.5.5 ADCn_SCANCTRL - Scan Control Register

Offset	Bit Position																													
0x010	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x0							0	0x0					0x0			0x00										0x0			0	0	0
Access	RW							RW	RW					RW			RW										RW			RW	RW	RW
Name	PRSSEL							PRSEN	AT					REF			INPUTMASK										RES			ADJ	DIFF	REP

Bit	Name	Reset	Access	Description
31:28	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

Bit	Name	Reset	Access	Description
27:25	Reserved	To ensure compatibility with future devices, always write bits to 0.		
24	PRSEN	0	RW	**Scan Sequence PRS Trigger Enable**

Enabled/disable PRS trigger of scan sequence.

Bit	Name	Reset	Access	Description

Value	Description
0	Scan sequence is not triggered by PRS input
1	Scan sequence is triggered by PRS input selected by PRSSEL

23:20 AT 0x0 RW **Scan Sample Acquisition Time**

Select the acquisition time for scan samples.

Value	Mode	Description
0	1CYCLE	1 ADC_CLK cycle acquisition time for scan samples
1	2CYCLES	2 ADC_CLK cycles acquisition time for scan samples
2	4CYCLES	4 ADC_CLK cycles acquisition time for scan samples
3	8CYCLES	8 ADC_CLK cycles acquisition time for scan samples
4	16CYCLES	16 ADC_CLK cycles acquisition time for scan samples
5	32CYCLES	32 ADC_CLK cycles acquisition time for scan samples
6	64CYCLES	64 ADC_CLK cycles acquisition time for scan samples
7	128CYCLES	128 ADC_CLK cycles acquisition time for scan samples
8	256CYCLES	256 ADC_CLK cycles acquisition time for scan samples

19 Reserved To ensure compatibility with future devices, always write bits to 0.

18:16 REF 0x0 RW **Scan Sequence Reference Selection**

Select reference to ADC scan sequence.

Value	Mode	Description
0	1V25	Internal 1.25 V reference
1	2V5	Internal 2.5 V reference
2	VDD	VDD
3	5VDIFF	Internal differential 5 V reference
4	EXTSINGLE	Single ended external reference from pin 6
5	2XEXTDIFF	Differential external reference, 2x(pin 6 - pin 7)
6	2XVDD	Unbuffered 2xVDD

15:8 INPUTMASK 0x00 RW **Scan Sequence Input Mask**

Set one or more bits in this mask to select which inputs are included the scan sequence in either single ended or differential mode.

DIFF = 0		
Mode	Value	Description
CH0	00000001	ADCn_CH0 included in mask
CH1	00000010	ADCn_CH1 included in mask
CH2	00000100	ADCn_CH2 included in mask
CH3	00001000	ADCn_CH3 included in mask
CH4	00010000	ADCn_CH4 included in mask
CH5	00100000	ADCn_CH5 included in mask
CH6	01000000	ADCn_CH6 included in mask
CH7	10000000	ADCn_CH7 included in mask
DIFF = 1		
Mode	Value	Description
CH0CH1	00000001	(Positive input: ADCn_CH0 Negative input: ADCn_CH1) included in mask
CH2CH3	00000010	(Positive input: ADCn_CH2 Negative input: ADCn_CH3) included in mask
CH4CH5	00000100	(Positive input: ADCn_CH4 Negative input: ADCn_CH5) included in mask
CH6CH7	00001000	(Positive input: ADCn_CH6 Negative input: ADCn_CH7) included in mask
	0001xxxx-1111xxxx	Reserved

7:6 Reserved To ensure compatibility with future devices, always write bits to 0.

5:4 RES 0x0 RW **Scan Sequence Resolution Select**

Select scan sequence conversion resolution.

Bit	Name	Reset	Access	Description
	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
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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		
	1	Scan conversion mode is converting continuously until SCANSTOP is written		

AB.5.6 ADCn_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
Access																							RW	RW							RW	RW
Name																							SCANOF	SINGLEOF							SCAN	SINGLE

Bit	Name	Reset	Access	Description
31:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9	SCANOF	0	RW	**Scan Result Overflow Interrupt Enable**
	Enable/disable scan result overflow interrupt.			
8	SINGLEOF	0	RW	**Single Result Overflow Interrupt Enable**
	Enable/disable single result overflow interrupt.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	SCAN	0	RW	**Scan Conversion Complete Interrupt Enable**
	Enable/disable scan conversion complete interrupt.			
0	SINGLE	0	RW	**Single Conversion Complete Interrupt Enable**
	Enable/disable single conversion complete interrupt.			

AB.5.7 ADCn_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
Access																							R	R							R	R

Offset	Bit Position																													
Name																						SCANOF	SINGLEOF						SCAN	SINGLE

Bit	Name	Reset	Access	Description
31:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9	SCANOF	0	R	**Scan Result Overflow Interrupt Flag**
	Indicates scan result overflow when this bit is set.			
8	SINGLEOF	0	R	**Single Result Overflow Interrupt Flag**
	Indicates single result overflow when this bit is set.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	SCAN	0	R	**Scan Conversion Complete Interrupt Flag**
	Indicates scan conversion complete when this bit is set.			
0	SINGLE	0	R	**Single Conversion Complete Interrupt Flag**
	Indicates single conversion complete when this bit is set.			

AB.5.8 ADCn_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
Access																							W1	W1							W1	W1
Name																							SCANOF	SINGLEOF							SCAN	SINGLE

Bit	Name	Reset	Access	Description
31:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9	SCANOF	0	W1	**Scan Result Overflow Interrupt Flag Set**
	Write to 1 to set scan result overflow interrupt flag			
8	SINGLEOF	0	W1	**Single Result Overflow Interrupt Flag Set**
	Write to 1 to set single result overflow interrupt flag.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	SCAN	0	W1	**Scan Conversion Complete Interrupt Flag Set**
	Write to 1 to set scan conversion complete interrupt flag.			
0	SINGLE	0	W1	**Single Conversion Complete Interrupt Flag Set**
	Write to 1 to set single conversion complete interrupt flag.			

AB.5.9 ADCn_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
Access																							W1	W1							W1	W1
Name																							SCANOF	SINGLEOF							SCAN	SINGLE

Bit	Name	Reset	Access	Description
31:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9	SCANOF	0	W1	**Scan Result Overflow Interrupt Flag Clear**
	Write to 1 to clear scan result overflow interrupt flag.			
8	SINGLEOF	0	W1	**Single Result Overflow Interrupt Flag Clear**
	Write to 1 to clear single result overflow interrupt flag.			
7:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	SCAN	0	W1	**Scan Conversion Complete Interrupt Flag Clear**
	Write to 1 to clear scan conversion complete interrupt flag.			
0	SINGLE	0	W1	**Single Conversion Complete Interrupt Flag Clear**
	Write to 1 to clear single conversion complete interrupt flag.			

AB.5.10 ADCn_SINGLEDATA - Single Conversion Result Data

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	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	R	**Single Conversion Result Data**
	The register holds the results from the last single conversion. Reading this field clears the SINGLEDV bit in the ADCn_STATUS register.			

AB.5.11 ADCn_SCANDATA - Scan Conversion Result Data

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	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	R	**Scan Conversion Result Data**
	The register holds the results from the last scan conversion. Reading this field clears the SCANDV bit in the ADCn_STATUS register.			

AB.5.12 ADCn_SINGLEDATAP - Single Conversion Result Data Peek Register

Offset	Bit Position																														
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00000000																															
Access	R																															
Name	DATAP																															

Bit	Name	Reset	Access	Description
31:0	DATAP	0x00000000	R	**Single Conversion Result Data Peek**

Bit	Name	Reset	Access	Description
	The register holds the results from the last single conversion. Reading this field will not clear SINGLEDV in ADCn_STATUS or SINGLE DMA request.			

AB.5.13 ADCn_SCANDATAP - Scan Sequence Result Data Peek Register

Offset	Bit Position																													
0x030	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	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 conversion. Reading this field will not clear SCANDV in ADCn_STATUS or single DMA request.			

AB.5.14 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		0x3F								0x00								0x3F								0x00						
Access		RW								RW								RW								RW						
Name		SCANGAIN								SCANOFFSET								SINGLEGAIN								SINGLEOFFSET						

Bit	Name	Reset	Access	Description
31	Reserved	To ensure compatibility with future devices, always write bits to 0.		
30:24	SCANGAIN	0x3F	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	Reserved	To ensure compatibility with future devices, always write bits to 0.		
22:16	SCANOFFSET	0x00	RW	**Scan Mode Offset Calibration Value**
	This register contains the offset calibration value used with scan conversions. 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	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14:8	SINGLEGAIN	0x3F	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	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:0	SINGLEOFFSET	0x00	RW	**Single Mode Offset Calibration Value**
	This register contains the offset calibration value used with single conversions. 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.			

AB.5.15 ADCn_BIASPROG - Bias Programming 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																					0x7					1			0x7			
Access																					RW					RW			RW			
Name																					COMPBIAS					HALFBIAS			BIASPROG			

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11:8	COMPBIAS	0x7	RW	**Comparator Bias Value**
	These bits are used to adjust the bias current to the ADC Comparator.			
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6	HALFBIAS	1	RW	**Half Bias Current**
	Set this bit to halve the bias current.			
5:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3:0	BIASPROG	0x7	RW	**Bias Programming Value**
	These bits are used to adjust the bias current.			

AC ARM Digital to Analog Converter

AC.1 Introduction

The Digital to Analog Converter (DAC) can convert a digital value to an analog output voltage. The DAC 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 DAC may be used for a number of different applications such as sensor interfaces or sound output.

AC.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 2.5V
  - Internal 1.25V
  - V_{DD}
- Conversion triggers
  - Data write
  - PRS input
- Automatic refresh timer
  - Selection from 16-64 prescaled HFPERCLK cycles
  - Individual refresh enable for each channel
- Interrupt generation on finished conversion
  - Separate interrupt flag for each channel
- PRS output pulse on finished conversion
  - Separate line for each channel
- DMA request on finished conversion
  - Separate request for each channel
- Support for offset and gain calibration
- Output to ADC
- Sine generation mode

- Optional high strength line driver

AC.3 Functional Description

An overview of the DAC module is shown in Figure 284.

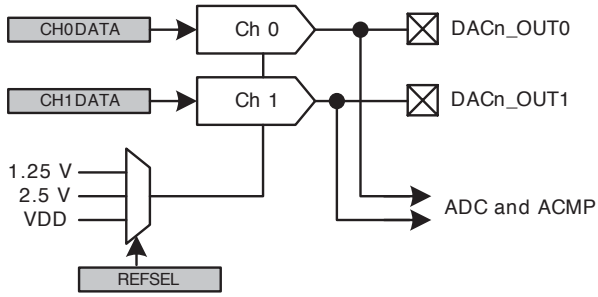


Figure 284: DAC Overview

AC.3.1 Conversions

The DAC consists of two channels (Channel 0 and 1) with separate 12-bit data registers (DACn_CH0DATA and DACn_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 DAC supports three conversion modes, continuous, sample/hold, sample/off.

Continuous Mode

In continuous mode the DAC channels will drive their outputs continuously with the data in the DACn_CHxDATA registers. This mode will maintain the output voltage and refresh is therefore not needed.

Sample/Hold Mode

In sample/hold mode, the DAC core converts data on a triggered conversion and then holds the output in a sample/hold element. When not converting, the DAC core is turned off between samples, which reduces the power consumption. Because of output voltage drift the sample/hold element will only hold the output for a certain period without a refresh conversion. The reader is referred to the electrical characteristics for the details on the voltage drift. The sampling period in this mode is set to the length of one prescaled clock cycle.

Sample/Off Mode

In sample/off mode the DAC and the sample/hold element is turned completely off between samples, tri-stating the DAC output. This requires the DAC output voltage to be held externally. The references are also turned off between samples, which means that a new warm-up period is needed before each conversion. The sampling period in this mode is set to the length of one prescaled clock cycle.

Conversion Start

The DAC channel must be enabled before it can be used. When the channel is enabled, a conversion can be started by writing to the DACn_CHxDATA register. These data registers are also mapped into a combined data register, DACn_COMBDATA, where the data values for both channels can be written simultaneously. Writing to this register will start all enabled channels.

If the PRSEN bit in DACn_CHxCTRL is set, a DAC conversion on channel x will not be started by data write, but when a positive one HFPERCLK cycle pulse is received on the PRS input selected by PRSSEL in DACn_CHxCTRL.

The CH0DV and CH1DV bits in DACn_STATUS indicate that the corresponding channel contains data that has not yet been converted.

When entering Energy Mode 4, both DAC channels must be stopped.

Clock Prescaling

The DAC has an internal clock prescaler, which can divide the HFPERCLK by any factor between 1 and 128, by setting the PRESC bits in DACnCTRL. The resulting DAC_CLK is used by the converter core and the frequency is given by Equation 41 :

$$f_{\text{DAC\_CLK}} = f_{\text{HFPERCLK}} / 2^{PRESC} \quad (41)$$

where f_{HFPERCLK} is the HFPERCLK frequency. One conversion takes 2 DAC_CLK cycles and the DAC_CLK should not be set higher than 1 MHz.

Normally 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 is controlled by the prescaled clock and the conversion can arrive at any time during a prescaled clock (DAC_CLK) period. However, if the CH0PRESCRST bit in DACn_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.

AC.3.2 Reference Selection

Three internal voltage references are available and are selected by setting the REFSEL bits in DACn_CTRL:

- Internal 2.5V
- Internal 1.25V
- V_{DD}

The reference selection can only be changed while both channels are disabled. The references for the DAC need to be enabled for some time before they can be used. This is called the warm-up period, and starts when one of the channels is enabled. For a bandgap reference, this period is 5 DAC_CLK cycles while the V_{DD} reference

needs 1 DAC_CLK cycle. The DAC will time this period automatically(given that the prescaler is set correctly) and delay any conversion triggers received during the warm-up until the references have stabilized.

AC.3.3 Programming of Bias Current

The bias current of the bandgap reference and the DAC output buffer can be scaled by the BIASPROG and HALFBIAS bit fields of the DACn_BIASPROG register as illustrated in Figure 285.

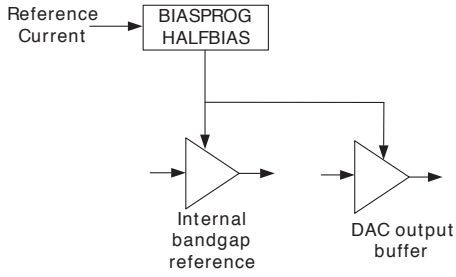


Figure 285: DAC Bias Programming

The minimum value of the BIASPROG bit-field of the DACn_BIASPROG register (i.e. BIASPROG=0b0000) represents the minimum bias current. Similarly BIASPROG=0b1111 represents the maximum bias current. The bias current defined by the BIASPROG setting can be halved by setting the HALFBIAS bit of the DACn_BIASPROG register.

The bias current settings should only be changed while both DAC channels are disabled. The electrical characteristics given in the datasheet require the bias configuration to be set to the default values, where no other bias values are given.

AC.3.4 Mode

The two DAC channels can act as two separate single ended channels or be combined into one differential channel. This is selected through the DIFF bit in DACn_CTRL.

Single Ended Output

When operating in single ended mode, the channel 0 output is on DACn_OUT0 and the channel 1 output is on DACn_OUT1. The output voltage can be calculated using Equation 42

$$V_{\text{OUT}} = V_{\text{DACn\_OUTx}} - V_{\text{SS}} = V_{\text{ref}} x CHx DATA / 4095 \quad (42)$$

where CHxDATA is a 12-bit unsigned integer.

Differential Output

When operating in differential mode, both DAC outputs are used as output for the bipolar voltage. The differential conversion uses DACn_CH0DATA as source. The positive output is on DACn_OUT1 and the negative output is on DACn_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 Equation 43:

$$V_{\text{OUT}} = V_{\text{DACn\_OUT1}} - V_{\text{DACn\_OUT0}} = V_{\text{ref}} \times CH0DATA/2047 \quad (43)$$

where CH0DATA is a 12-bit signed integer. The common mode voltage is $V_{DD}/2$.

AC.3.5 Sine Generation Mode

The DAC contains an automatic sine-generation mode, which is enabled by setting the SINEMODE bit in DACn_CTRL. In this mode, the DAC 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 DACn_CH0CTRL. When the PRS line is low, a voltage of Vref/2 will be produced. 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 Equation 44:

$$f_{\text{sine}} = f_{\text{HFPERCLK}}/32x(PRESC + 1) \quad (44)$$

The SINE wave will be output on channel 0. If DIFF is set in DACn_CTRL, the sine wave will be output on both channels (if enabled), but inverted (see Figure 284). Note that when OUTENPRS in DACn_CTRL is set, the sine output will be reset to 0 degrees when the PRS line selected by CH1PRSSEL is low.

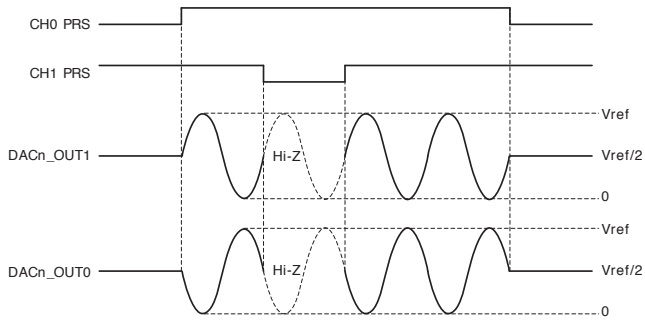


Figure 286: DAC Sine Mode

AC.3.6 Interrupts and PRS Output

Both DAC channels have separate interrupt flags (in DACn_IF) indicating that a conversion has finished on the channel and that new data can be written to the

data registers. Setting one of these flags will result in a DAC interrupt if the corresponding interrupt enable bit is set in DACn_IEN. All generated interrupts from the DAC will activate the same interrupt vector when enabled.

The DAC has two PRS outputs which will carry a one cycle (HFPERCLK) high pulse when the corresponding channel has finished a conversion.

AC.3.7 DMA Request

The DAC sends out a DMA request when a conversion on a channel is complete. This request is cleared when the corresponding channel's data register is written.

AC.3.8 Analog Output

Each DAC channel has its own output pin (DACn_OUT0 and DACn_OUT1) in addition to an internal loopback to the ADC and ACMP. These outputs can be enabled and disabled individually in the EN field in DACn_CHxCTRL registers in combination with OUTPUTSEL in DACn_CTRL. The DAC outputs can also be directed to the ADC and ACMP, which is also configurable in the OUTPUTSEL field in DACn_CTRL.

The DAC outputs are tri-stated when the channels are not enabled. By setting the OUTENPRS bit in DACn_CTRL, the outputs are also tri-stated when the PRS line selected by CH1PRSSEL in DACn_CH1CTRL is low. When the PRS signal is high, the outputs are enabled as normal.

The DAC channels can also drive an alternative output network, which is described in the Opamp chapter. To enable this network, OUTMODE must be configured to ADC in DACn_CTRL. The actual output network can be configred by configuring DACn_OPAxMUX registers.

AC.3.9 Calibration

The DAC contains a calibration register, DACn_CAL, where calibration values for both offset and gain correction can be written. Offset calibration is done separately for each channel through the CHxOFFSET bit-fields. Gain is calibrated in one common register field, GAIN. The gain calibration is linked to the reference and when the reference is changed, the gain must be re-calibrated. Gain and offset for the 1V25, 2V5 and VDD references 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.

AC.3.10 Opamps

The DAC includes a set of three highly configurable opamps that can be accessed in the DAC module. Two of the opamps are located in the DAC, while the third

opamp is a standalone opamp. For detailed description see the OPAMP chapter. The register description can be found Section AC.5

AC.4 Register Map

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

Offset	Name	Type	Description
0x000	DACn_CTRL	RW	Control Register
0x004	DACn_STATUS	R	Status Register
0x008	DACn_CH0CTRL	RW	Channel 0 Control Register
0x00C	DACn_CH1CTRL	RW	Channel 1 Control Register
0x010	DACn_IEN	RW	Interrupt Enable Register
0x014	DACn_IF	R	Interrupt Flag Register
0x018	DACn_IFS	W1	Interrupt Flag Set Register
0x01C	DACn_IFC	W1	Interrupt Flag Clear Register
0x020	DACn_CH0DATA	RW	Channel 0 Data Register
0x024	DACn_CH1DATA	RW	Channel 1 Data Register
0x028	DACn_COMBDATA	W	Combined Data Register
0x02C	DACn_CAL	RW	Calibration Register
0x030	DACn_BIASPROG	RW	Bias Programming Register
0x054	DACn_OPACTRL	RW	Operational Amplifier Control Register
0x058	DACn_OPAOFFSET	RW	Operational Amplifier Offset Register
0x05C	DACn_OPA0MUX	RW	Operational Amplifier Mux Configuration Register
0x060	DACn_OPA1MUX	RW	Operational Amplifier Mux Configuration Register
0x064	DACn_OPA2MUX	RW	Operational Amplifier Mux Configuration Register

AC.5 Register Description

AC.5.1 DACn_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									0x0		0	0	0x1		0x0		0	0
Access											RW			RW									RW		RW	RW	RW		RW		RW	RW

Offset	Bit Position															
Name			REFRSEL		PRESC		REFSEL	CH0PRESCRST	OUTENPRS	OUTMODE	CONVMODE	SINEMODE	DIFF

Bit	Name	Reset	Access	Description
31:22	Reserved	To ensure compatibility with future devices, always write bits to 0.		
21:20	REFRSEL	0x0	RW	**Refresh Interval Select**

Select refresh counter timeout value. A channel x will be refreshed with the interval set in this register if the REFREN bit in DACn_CHxCTRL is set.

Value	Mode	Description
0	8CYCLES	All channels with enabled refresh are refreshed every 8 prescaled cycles
1	16CYCLES	All channels with enabled refresh are refreshed every 16 prescaled cycles
2	32CYCLES	All channels with enabled refresh are refreshed every 32 prescaled cycles
3	64CYCLES	All channels with enabled refresh are refreshed every 64 prescaled cycles

19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18:16	PRESC	0x0	RW	**Prescaler Setting**

Select clock division factor.

Value	Description
PRESC	Clock division factor of 2^PRESC.

15:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9:8	REFSEL	0x0	RW	**Reference Selection**

Select reference.

Value	Mode	Description
0	1V25	Internal 1.25 V bandgap reference
1	2V5	Internal 2.5 V bandgap reference
2	VDD	VDD reference

7	CH0PRESCRST	0	RW	**Channel 0 Start Reset Prescaler**

Select if prescaler is reset on channel 0 start.

Value	Description
0	Prescaler not reset on channel 0 start
1	Prescaler reset on channel 0 start

6	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.

5:4	OUTMODE	0x1	RW	**Output Mode**

Select output mode.

Value	Mode	Description
0	DISABLE	DAC output to pin and ADC disabled
1	PIN	DAC output to pin enabled. DAC output to ADC and ACMP disabled
2	ADC	DAC output to pin disabled. DAC output to ADC and ACMP enabled
3	PINADC	DAC output to pin, ADC, and ACMP enabled

3:2	CONVMODE	0x0	RW	**Conversion Mode**

Configure conversion mode.

Value	Mode	Description
0	CONTINUOUS	DAC is set in continuous mode
1	SAMPLEHOLD	DAC is set in sample/hold mode
2	SAMPLEOFF	DAC is set in sample/shut off mode

1	SINEMODE	0	RW	**Sine Mode**

Enable/disable sine mode.

Bit	Name	Reset	Access	Description
	Value	**Description**		
	0	Sine mode disabled. Sine reset to 0 degrees		
	1	Sine mode enabled		
0	DIFF	0	RW	**Differential Mode**
	Select single ended or differential mode.			
	Value	**Description**		
	0	Single ended output		
	1	Differential output		

AC.5.2 DACn_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
Access																															R	R
Name																															CH1DV	CH0DV

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	CH1DV	0	R	**Channel 1 Data Valid**
	This bit is set high when CH1DATA is written and is set low when CH1DATA is used in conversion.			
0	CH0DV	0	R	**Channel 0 Data Valid**
	This bit is set high when CH0DATA is written and is set low when CH0DATA is used in conversion.			

AC.5.3 DACn_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	0	0
Access																									RW					RW	RW	RW
Name																									PRSSEL					PRSEN	REFREN	EN

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:4	PRSSEL	0x0	RW	**Channel 0 PRS Trigger Select**
	Select Channel 0 PRS input channel.			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	PRSCH0	PRS ch 0 triggers channel 0 conversion.
1	PRSCH1	PRS ch 1 triggers channel 0 conversion.
2	PRSCH2	PRS ch 2 triggers channel 0 conversion.
3	PRSCH3	PRS ch 3 triggers channel 0 conversion.
4	PRSCH4	PRS ch 4 triggers channel 0 conversion.
5	PRSCH5	PRS ch 5 triggers channel 0 conversion.
6	PRSCH6	PRS ch 6 triggers channel 0 conversion.
7	PRSCH7	PRS ch 7 triggers channel 0 conversion.
8	PRSCH8	PRS ch 8 triggers channel 0 conversion.
9	PRSCH9	PRS ch 9 triggers channel 0 conversion.
10	PRSCH10	PRS ch 10 triggers channel 0 conversion.
11	PRSCH11	PRS ch 11 triggers channel 0 conversion.

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	PRSEN	0	RW	**Channel 0 PRS Trigger Enable**

Select Channel 0 conversion trigger.

Value	Description
0	Channel 0 is triggered by CH0DATA or COMBDATA write
1	Channel 0 is triggered by PRS input

Bit	Name	Reset	Access	Description
1	REFREN	0	RW	**Channel 0 Automatic Refresh Enable**

Set to enable automatic refresh of channel 0. Refresh period is set by REFRSEL in DACn_CTRL.

Value	Description
0	Channel 0 is not refreshed automatically
1	Channel 0 is refreshed automatically

Bit	Name	Reset	Access	Description
0	EN	0	RW	**Channel 0 Enable**

Enable/disable channel 0.

AC.5.4 DACn_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	0	0
Access																									RW					RW	RW	RW
Name																									PRSSEL					PRSEN	REFREN	EN

Bit	Name	Reset	Access	Description
31:8	Reserved	To ensure compatibility with future devices, always write bits to 0.		
7:4	PRSSEL	0x0	RW	**Channel 1 PRS Trigger Select**

Select Channel 1 PRS input channel.

Bit	Name	Reset	Access	Description
	Value	**Mode**		**Description**
	0	PRSCH0		PRS ch 0 triggers channel 1 conversion.
	1	PRSCH1		PRS ch 1 triggers channel 1 conversion.
	2	PRSCH2		PRS ch 2 triggers channel 1 conversion.
	3	PRSCH3		PRS ch 3 triggers channel 1 conversion.
	4	PRSCH4		PRS ch 4 triggers channel 1 conversion.
	5	PRSCH5		PRS ch 5 triggers channel 1 conversion.
	6	PRSCH6		PRS ch 6 triggers channel 1 conversion.
	7	PRSCH7		PRS ch 7 triggers channel 1 conversion.
	8	PRSCH8		PRS ch 8 triggers channel 1 conversion.
	9	PRSCH9		PRS ch 9 triggers channel 1 conversion.
	10	PRSCH10		PRS ch 10 triggers channel 1 conversion.
	11	PRSCH11		PRS ch 11 triggers channel 1 conversion.
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	PRSEN	0	RW	**Channel 1 PRS Trigger Enable**
	Select Channel 1 conversion trigger.			
	Value	**Description**		
	0	Channel 1 is triggered by CH1DATA or COMBDATA write		
	1	Channel 1 is triggered by PRS input		
1	REFREN	0	RW	**Channel 1 Automatic Refresh Enable**
	Set to enable automatic refresh of channel 1. Refresh period is set by REFRSEL in DACn_CTRL.			
	Value	**Description**		
	0	Channel 1 is not refreshed automatically		
	1	Channel 1 is refreshed automatically		
0	EN	0	RW	**Channel 1 Enable**
	Enable/disable channel 1.			

AC.5.5 DACn_IEN - Interrupt Enable 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
Access																											RW	RW			RW	RW
Name																											CH1UF	CH0UF			CH1	CH0

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5	CH1UF	0	RW	**Channel 1 Conversion Data Underflow Interrupt Enable**
	Enable/disable channel 1 data underflow interrupt.			
4	CH0UF	0	RW	**Channel 0 Conversion Data Underflow Interrupt Enable**
	Enable/disable channel 0 data underflow interrupt.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	CH1	0	RW	**Channel 1 Conversion Complete Interrupt Enable**
	Enable/disable channel 1 conversion complete interrupt.			
0	CH0	0	RW	**Channel 0 Conversion Complete Interrupt Enable**
	Enable/disable channel 0 conversion complete interrupt.			

AC.5.6 DACn_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
Access																											R	R			R	R
Name																											CH1UF	CH0UF			CH1	CH0

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	CH1	0	R	**Channel 1 Conversion Complete Interrupt Flag**
	Indicates channel 1 conversion complete and that new data can be written to the data register.			
0	CH0	0	R	**Channel 0 Conversion Complete Interrupt Flag**
	Indicates channel 0 conversion complete and that new data can be written to the data register.			

AC.5.7 DACn_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
Access																											W1	W1			W1	W1
Name																											CH1UF	CH0UF			CH1	CH0

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5	CH1UF	0	W1	**Channel 1 Data Underflow Interrupt Flag Set**
	Write to 1 to set channel 1 Data Underflow interrupt flag.			
4	CH0UF	0	W1	**Channel 0 Data Underflow Interrupt Flag Set**
	Write to 1 to set channel 0 Data Underflow interrupt flag.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	CH1	0	W1	**Channel 1 Conversion Complete Interrupt Flag Set**
	Write to 1 to set channel 1 conversion complete interrupt flag.			
0	CH0	0	W1	**Channel 0 Conversion Complete Interrupt Flag Set**
	Write to 1 to set channel 0 conversion complete interrupt flag.			

AC.5.8 DACn_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
Access																											W1	W1			W1	W1
Name																											CH1UF	CH0UF			CH1	CH0

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5	CH1UF	0	W1	**Channel 1 Data Underflow Interrupt Flag Clear**
	Write to 1 to clear channel 1 data underflow interrupt flag.			
4	CH0UF	0	W1	**Channel 0 Data Underflow Interrupt Flag Clear**
	Write to 1 to clear channel 0 data underflow interrupt flag.			
3:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	CH1	0	W1	**Channel 1 Conversion Complete Interrupt Flag Clear**
	Write to 1 to clear channel 1 conversion complete interrupt flag.			
0	CH0	0	W1	**Channel 0 Conversion Complete Interrupt Flag Clear**
	Write to 1 to clear channel 0 conversion complete interrupt flag.			

AC.5.9 DACn_CH0DATA - Channel 0 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																					0x000											
Access																					RW											
Name																					DATA											

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11:0	DATA	0x000	RW	**Channel 0 Data**
	This register contains the value which will be converted by channel 0.			

AC.5.10 DACn_CH1DATA - Channel 1 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																					0x000											
Access																					RW											
Name																					DATA											

Bit	Name	Reset	Access	Description
31:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11:0	DATA	0x000	RW	**Channel 1 Data**
	This register contains the value which will be converted by channel 1.			

AC.5.11 DACn_COMBDATA - Combined 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					0x000																0x000											
Access					W																W											

Offset	Bit Position																													
Name												CH1DATA																	CH0DATA		

Bit	Name	Reset	Access	Description
31:28	Reserved	To ensure compatibility with future devices, always write bits to 0.		
27:16	CH1DATA	0x000	W	**Channel 1 Data**
	Data written to this register will be written to DATA in DACn_CH1DATA.			
15:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11:0	CH0DATA	0x000	W	**Channel 0 Data**
	Data written to this register will be written to DATA in DACn_CH0DATA.			

AC.5.12 DACn_CAL - Calibration Register

Offset	Bit Position																													
0x02C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset										0x40									0x00								0x00					
Access										RW									RW								RW					
Name										GAIN									CH1OFFSET								CH0OFFSET					

Bit	Name	Reset	Access	Description
31:23	Reserved	To ensure compatibility with future devices, always write bits to 0.		
22:16	GAIN	0x40	RW	**Gain Calibration Value**
	This register contains the gain calibration value. 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 lower DAC results.			
15:14	Reserved	To ensure compatibility with future devices, always write bits to 0.		
13:8	CH1OFFSET	0x00	RW	**Channel 1 Offset Calibration Value**
	This register contains the offset calibration value used with channel 1 conversions. This field is set to the production channel 1 offset calibration value for the 1V25 internal reference during reset, hence the reset value might differ from device to device. The field is sign-magnitude encoded. Higher values lead to lower DAC results.			
7:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:0	CH0OFFSET	0x00	RW	**Channel 0 Offset Calibration Value**
	This register contains the offset calibration value used with channel 0 conversions. This field is set to the production channel 0 offset calibration value for the 1V25 internal reference during reset, hence the reset value might differ from device to device. The field is sign-magnitude encoded. Higher values lead to lower DAC results.			

AC.5.13 DACn_BIASPROG - Bias Programming 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																		1			0x7					1			0x7			
Access																		RW			RW					RW			RW			
Name																		OPA2HALFBIAS			OPA2BIASPROG					HALFBIAS			BIASPROG			

Bit	Name	Reset	Access	Description
31:15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14	OPA2HALFBIAS	1	RW	**Half Bias Current**
	Set this bit to halve the bias current.			
13:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11:8	OPA2BIASPROG	0x7	RW	**Bias Programming Value for OPA2**
	These bits control the bias current level.			
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6	HALFBIAS	1	RW	**Half Bias Current**
	Set this bit to halve the bias current.			
5:4	Reserved	To ensure compatibility with future devices, always write bits to 0.		
3:0	BIASPROG	0x7	RW	**Bias Programming Value**
	These bits control the bias current level.			

AC.5.14 DACn_OPACTRL - Operational Amplifier 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	0	0					0x0		0x0		0x0					0	0	0				0	0	0
Access								RW	RW	RW					RW		RW		RW					RW	RW	RW				RW	RW	RW
Name								OPA2SHORT	OPA1SHORT	OPA0SHORT					OPA2LPFDIS		OPA1LPFDIS		OPA0LPFDIS					OPA2HCMDIS	OPA1HCMDIS	OPA0HCMDIS				OPA2EN	OPA1EN	OPA0EN

Bit	Name	Reset	Access	Description
31:25	Reserved	To ensure compatibility with future devices, always write bits to 0.		
24	OPA2SHORT	0	RW	**Short the non-inverting and inverting input.**
	Set to short the non-inverting and inverting input.			
23	OPA1SHORT	0	RW	**Short the non-inverting and inverting input.**
	Set to short the non-inverting and inverting input.			
22	OPA0SHORT	0	RW	**Short the non-inverting and inverting input.**
	Set to short the non-inverting and inverting input.			
21:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17:16	OPA2LPFDIS	0x0	RW	**Disables Low Pass Filter.**
	Disables the low pass filter between pad and the positive and negative input mux.			

LPF DISABLE	VALUE	Description
PLPFDIS	x1	Disables the low pass filter between positive pad and positive input.
NLPFDIS	1x	Disables the low pass filter between negative pad and negative input.

15:14	OPA1LPFDIS	0x0	RW	**Disables Low Pass Filter.**
	Disables the low pass filter between pad and the positive and negative input mux.			

LPF DISABLE	VALUE	Description
PLPFDIS	x1	Disables the low pass filter between positive pad and positive input.
NLPFDIS	1x	Disables the low pass filter between negative pad and negative input.

13:12	OPA0LPFDIS	0x0	RW	**Disables Low Pass Filter.**
	Disables the low pass filter between pad and the positive and negative input mux.			

LPF DISABLE	VALUE	Description
PLPFDIS	x1	Disables the low pass filter between positive pad and positive input.
NLPFDIS	1x	Disables the low pass filter between negative pad and negative input.

Bit	Name	Reset	Access	Description
11:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8	OPA2HCMDIS	0	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.			
7	OPA1HCMDIS	0	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.			
6	OPA0HCMDIS	0	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.			
5:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	OPA2EN	0	RW	**OPA2 Enable**
	Set to enable OPA2, clear to disable.			
1	OPA1EN	0	RW	**OPA1 Enable**
	Set to enable OPA1, clear to disable. CH1EN in DAC_CH1CTRL must also be set.			
0	OPA0EN	0	RW	**OPA0 Enable**
	Set to enable OPA0, clear to disable. CH0EN in DAC_CH0CTRL must also be set.			

AC.5.15 DACn_OPAOFFSET - Operational Amplifier Offset 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																											0x20					
Access																											RW					
Name																											OPA2OFFSET					

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:0	OPA2OFFSET	0x20	RW	**OPA2 Offset Configuration Value**
	This register contains the offset calibration value for OPA2. This field is set to the production OPA2 offset calibration value, hence the reset value might differ from device to device. The field is sign-magnitude encoded. Higher values lead to lower OPA results. The resolution of the LSB is 1.6mV/LSB			

AC.5.16 DACn_OPA0MUX - Operational Amplifier Mux Configuration Register

Offset	Bit Position																															
0x05C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0				0			0x1					0x00					0	0		0x0					0x0			0x0		
Access		RW				RW			RW					RW					RW	RW		RW					RW			RW		
Name		RESSEL				NEXTOUT			OUTMODE					OUTPEN					NPEN	PPEN		RESINMUX					NEGSEL			POSSEL		

Bit	Name	Reset	Access	Description
31	Reserved	To ensure compatibility with future devices, always write bits to 0.		
30:28	RESSEL	0x0	RW	**OPA0 Resistor Ladder Select**
	Configures the resistor ladder tap for OPA0.			

Bit	Name	Reset	Access	Description

Value	Mode	Resistor Value	Inverting Mode Gain (-R2/R1)	Non-inverting Mode Gain (1+(R2/R1)
0	RES0	R2 = 1/3 x R1	-1/3	1 1/3
1	RES1	R2 = R1	-1	2
2	RES2	R2 = 1 2/3 x R1	-1 2/3	2 2/3
3	RES3	R2 = 2 x R1	-2 1/5	3 1/5
4	RES4	R2 = 3 x R1	-3	4
5	RES5	R2 = 4 1/3 x R1	-4 1/3	5 1/3
6	RES6	R2 = 7 x R1	-7	8
7	RES7	R2 = 15 x R1	-15	16

Bit	Name	Reset	Access	Description
27	Reserved	To ensure compatibility with future devices, always write bits to 0.		
26	NEXTOUT	0	RW	**OPA0 Next Enable**
	Makes output of OPA0 available to OPA1.			
25:24	Reserved	To ensure compatibility with future devices, always write bits to 0.		
23:22	OUTMODE	0x1	RW	**Output Select**
	Select output channel.			

Value	Mode	Description
0	DISABLE	OPA0 output is disabled
1	MAIN	Main OPA0 output to pin enabled
2	ALT	OPA0 alternative output enabled.
3	ALL	Main OPA0 output drives both main and alternative outputs.

Bit	Name	Reset	Access	Description
21:19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18:14	OUTPEN	0x00	RW	**OPA0 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

Bit	Name	Reset	Access	Description
13	NPEN	0	RW	**OPA0 Negative Pad Input Enable**
	Connects pad to the negative input mux			
12	PPEN	0	RW	**OPA0 Positive Pad Input Enable**
	Connects pad to the positive input mux			
11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	RESINMUX	0x0	RW	**OPA0 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
1	OPA0INP	Set for OPA0 input
2	NEGPAD	NEG pad connected
3	POSPAD	POS pad connected
4	VSS	VSS connected

Bit	Name	Reset	Access	Description
7:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:4	NEGSEL	0x0	RW	**OPA0 inverting Input Mux**
	These bits selects the source for the inverting input on OPA0			

Value	Mode	Description
0	DISABLE	Input disabled
1	UG	Unity Gain feedback path
2	OPATAP	OPA0 Resistor ladder as input
3	NEGPAD	Input from NEG PAD

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	POSSEL	0x0	RW	**OPA0 non-inverting Input Mux**
	These bits selects the source for the non-inverting input on OPA0			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	DISABLE	Input disabled
1	DAC	DAC as input
2	POSPAD	POS PAD as input
3	OPA0INP	OPA0 as input
4	OPATAP	OPA0 Resistor ladder as input

AC.5.17 DACn_OPA1MUX - Operational Amplifier Mux Configuration Register

Offset	Bit Position																															
0x060	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0				0			0x0					0x00					0	0		0x0					0x0			0x0		
Access		RW				RW			RW					RW					RW	RW		RW					RW			RW		
Name		RESSEL				NEXTOUT			OUTMODE					OUTPEN					NPEN	PPEN		RESINMUX					NEGSEL			POSSEL		

Bit	Name	Reset	Access	Description
31	Reserved	To ensure compatibility with future devices, always write bits to 0.		
30:28	RESSEL	0x0	RW	**OPA1 Resistor Ladder Select**

Configures the resistor ladder tap for OPA1.

Value	Mode	Resistor Value	Inverting Mode Gain (-R2/R1)	Non-inverting Mode Gain (1+(R2/R1)
0	RES0	R2 = 1/3 x R1	-1/3	1 1/3
1	RES1	R2 = R1	-1	2
2	RES2	R2 = 1 2/3 x R1	-1 2/3	2 2/3
3	RES3	R2 = 2 x R1	-2 1/5	3 1/5
4	RES4	R2 = 3 x R1	-3	4
5	RES5	R2 = 4 1/3 x R1	-4 1/3	5 1/3
6	RES6	R2 = 7 x R1	-7	8
7	RES7	R2 = 15 x R1	-15	16

27	Reserved	To ensure compatibility with future devices, always write bits to 0.		
26	NEXTOUT	0	RW	**OPA1 Next Enable**

Makes output of OPA1 available to OPA2.

25:24	Reserved	To ensure compatibility with future devices, always write bits to 0.		
23:22	OUTMODE	0x0	RW	**Output Select**

Select output channel.

Value	Mode	Description
0	DISABLE	OPA0 output is disabled
1	MAIN	Main OPA1 output to pin enabled
2	ALT	OPA1 alternative output enabled.
3	ALL	Main OPA1 output drives both main and alternative outputs.

21:19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18:14	OUTPEN	0x00	RW	**OPA1 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

13	NPEN	0	RW	**OPA1 Negative Pad Input Enable**

Bit	Name	Reset	Access	Description
	Connects pad to the negative input mux			
12	PPEN	0	RW	**OPA1 Positive Pad Input Enable**
	Connects pad to the positive input mux			
11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	RESINMUX	0x0	RW	**OPA1 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
1	OPA0INP	Set for OPA0 input
2	NEGPAD	NEG PAD connected
3	POSPAD	POS PAD connected
4	VSS	VSS connected

Bit	Name	Reset	Access	Description
7:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:4	NEGSEL	0x0	RW	**OPA1 inverting Input Mux**
	These bits selects the source for the inverting input on OPA1			

Value	Mode	Description
0	DISABLE	Input disabled
1	UG	Unity Gain feedback path
2	OPATAP	OPA1 Resistor ladder as input
3	NEGPAD	Input from NEG PAD

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	POSSEL	0x0	RW	**OPA1 non-inverting Input Mux**
	These bits selects the source for the non-inverting input on OPA1			

Value	Mode	Description
0	DISABLE	Input disabled
1	DAC	DAC as input
2	POSPAD	POS PAD as input
3	OPA0INP	OPA0 as input
4	OPATAP	OPA 1 Resistor ladder as input

AC.5.18 DACn_OPA2MUX - Operational Amplifier Mux Configuration 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				0				0							0x0		0	0		0x0					0x0			0x0		
Access		RW				RW				RW							RW		RW	RW		RW					RW			RW		
Name		RESSEL				NEXTOUT				OUTMODE							OUTPEN		NPEN	PPEN		RESINMUX					NEGSEL			POSSEL		

Bit	Name	Reset	Access	Description
31	Reserved	To ensure compatibility with future devices, always write bits to 0.		
30:28	RESSEL	0x0	RW	**OPA2 Resistor Ladder Select**
	Configures the resistor ladder tap for OPA2.			

Bit	Name	Reset	Access	Description

Value	Mode	Resistor Value	Inverting Mode Gain (-R2/R1)	Non-inverting Mode Gain (1+(R2/R1)
0	RES0	R2 = 1/3 x R1	-1/3	1 1/3
1	RES1	R2 = R1	-1	2
2	RES2	R2 = 1 2/3 x R1	-1 2/3	2 2/3
3	RES3	R2 = 2 x R1	-2 1/5	3 1/5
4	RES4	R2 = 3 x R1	-3	4
5	RES5	R2 = 4 1/3 x R1	-4 1/3	5 1/3
6	RES6	R2 = 7 x R1	-7	8
7	RES7	R2 = 15 x R1	-15	16

Bit	Name	Reset	Access	Description
27	Reserved	To ensure compatibility with future devices, always write bits to 0.		
26	NEXTOUT	0	RW	**OPA2 Next Enable**
	OPA2 does not have an next output.			
25:23	Reserved	To ensure compatibility with future devices, always write bits to 0.		
22	OUTMODE	0	RW	**Output Select**
	Enables OPA2 main output.			
21:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:14	OUTPEN	0x0	RW	**OPA2 Output Location**
	Select location for main output			

Value	Mode	Description
1	OUT0	Main Output 0
2	OUT1	Main Output 1

Bit	Name	Reset	Access	Description
13	NPEN	0	RW	**OPA2 Negative Pad Input Enable**
	Connects pad to the negative input mux			
12	PPEN	0	RW	**OPA2 Positive Pad Input Enable**
	Connects pad to the positive input mux			
11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	RESINMUX	0x0	RW	**OPA2 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
1	OPA1INP	Set for OPA1 input
2	NEGPAD	NEG PAD connected
3	POSPAD	POS PAD connected
4	VSS	VSS connected

Bit	Name	Reset	Access	Description
7:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:4	NEGSEL	0x0	RW	**OPA2 inverting Input Mux**
	These bits selects the source for the inverting input on OPA2			

Value	Mode	Description
0	DISABLE	Input disabled
1	UG	Unity Gain feedback path
2	OPATAP	OPA2 Resistor ladder as input
3	NEGPAD	Input from NEG PAD

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	POSSEL	0x0	RW	**OPA2 non-inverting Input Mux**
	These bits selects the source for the non-inverting input on OPA2			

Value	Mode	Description
0	DISABLE	Input disabled
2	POSPAD	POS PAD as input
3	OPA1INP	OPA1 as input
4	OPATAP	OPA0 Resistor ladder as input

AD ARM Operational Amplifier

AD.1 Functional Description

The three opamps can be configured to perform various opamp functions through a network of muxes. An overview of the opamps are shown in Figure 287. Two of the three opamps are part of the DAC, while the third opamp is standalone. The output of OPA0 can be routed to ADC CH0, OPA1 and various pin outputs. The output of OPA1 can be routed to ADC CH1, OPA2, and various pin outputs. The output of OPA2 can be routed to ADC CH0, CH5, and various pin output destinations. All three opamps can also take input from pins. Since OPA0 and OPA1 are part of the DAC, special considerations needs to be taken when both the DAC Ch0/Ch1 and OPA0/OPA1 are being used. For detailed explanation the reader is referred to Section AD.1.3.

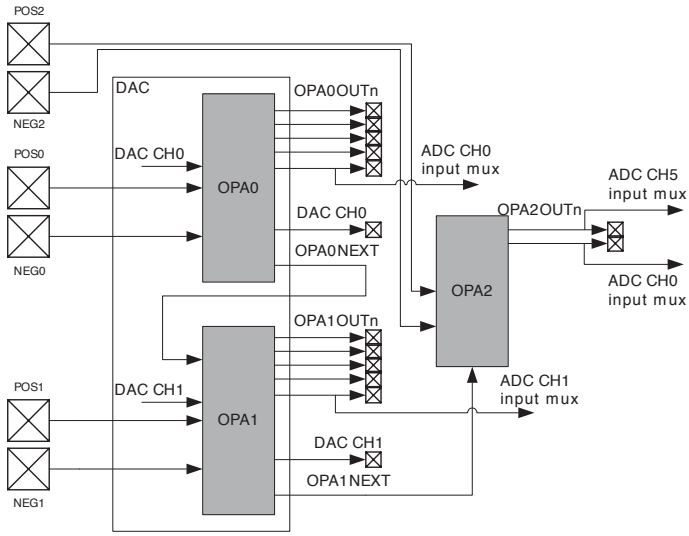


Figure 287: OPAMP System Overview

A more detailed view of the three opamps, including the mux network is shown in Figure 288. There is a set of input muxes for each opamp, making it possible to select various input sources. The POSSEL mux connected to the positive input makes it possible to select pin, another opamp output, or tap from the resistor network. Similarly, the NEGSEL mux on the negative input makes it possible to select pin or a feedback path as its source. The feedback path can be a unity gain configuration, or selected from the resistor network for programmable gain. The output of the opamp have different sets of outputs, a main output, an alternative output network and a next output. These outputs make it possible to route the output to pin, another opamp input, ADC, or into the feedback path. For details regarding configuring the outputs, the reader is referred to Section AD.1.1. In addition, there is also a mux to configure the resistor ladder to be connected to vss, pin, or another opamp output.

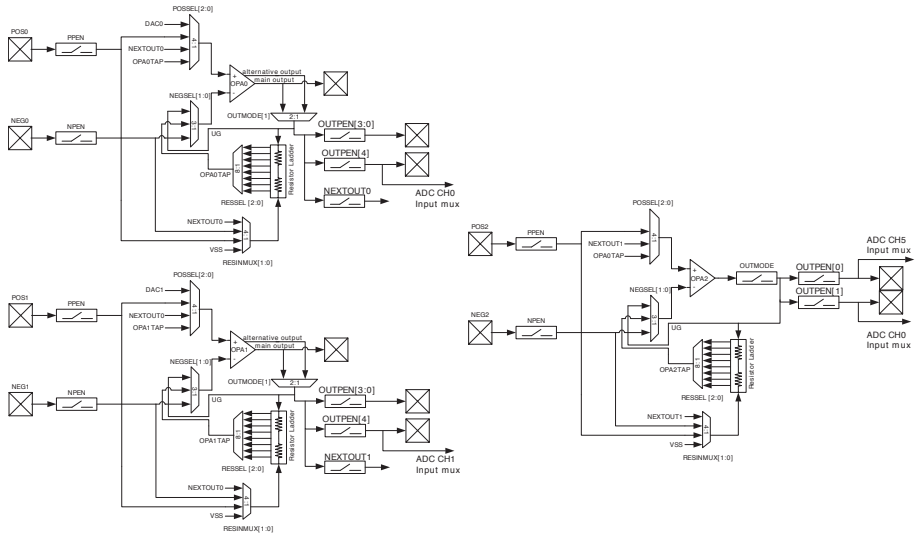


Figure 288: OPAMP Overview

AD.1.1 Opamp Configuration

Since two of the three opamps (OPA0, OPA1) are part of the DAC, the opamp configuration registers are located in the DAC. The mux registers for OPA0/OPA1 together with OPA2 registers are separate registers, also located under the DAC module. OPA0 and OPA1 can be enabled by setting OPAxEN in DACn_OPACTRL and CHxEN in CHxCTRL. OPA2 can be enabled by only setting OPA2EN in DACn_OPACTRL.

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 DACn_OPAxMUX register. Similarly, the mux connected to the negative input is configured by setting the NEGSEL bit-field in DACn_OPAxMUX. To connect the pins to the input muxes, the pin switches must also be enabled. Setting the PPEN bit-field enables to POSPADx, while setting the NPEN bit-field enables the NEGPADx, both located in DACn_OPAxMUX. The input into the resistor ladder can be configured by setting the RESINMUX bit-field in DACn_OPAxMUX.

Output Configuration

The opamp have two outputs, one main output and one alternative output with lower drive strength. These two outputs can be used to drive the different outputs as shown in Figure 289. The main opamp output can be used to drive the main output by setting OUTMODE to MAIN in DACn_OPAxMUX. The alternative opamp output can drive the alternative output network by setting OUTMODE to ALT in DACn_OPAxMUX. In addition, it is also possible to use the main opamp output to drive both the main output and the alternative output network by setting OUTMODE to ALL in DACn_OPAxMUX.

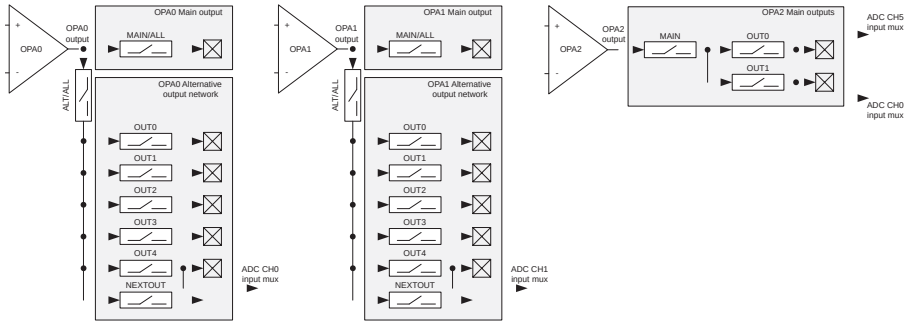


Figure 289: Opamp Output Stage Overview

The alternative output network consists of connections to pins, ADC, and a connection to the next opamp (OPA0 to OPA1, and OPA1 to OPA2). The connections to pins can be individually enabled by configuring OUTPEN in DACn_OPAxMUX register. To enable cascaded opamp configurations, each opamp has a NEXTOUT connection. This output makes it possible to connect OPA0 to OPA1, and OPA1 to OPA2. This output connection is enabled by setting NEXTOUT in DACn_OPAxMUX.

The opamps can also be routed to the ADC. OPA0 can be connected to ADC CH0, OPA1 to ADC CH1 and OPA2 can be connected to both ADC CH1 and CH5. The ADC connections are created by routing the OPA output by setting corresponding bits in OUTPEN in DACn_OPAxMUX. For OPA0 alternative output 4 is connected to ADC input mux CH0 when enabled. OPA1's alternative output 4 is connected to ADC input mux CH1 when enabled. For OPA2, the two main outputs can be connected to ADC input mux CH0 and ADC input mux CH5 respectively when enabled. See the ADC chapter for information on how to configure the ADC input mux.

Gain Programming

The feedback path of each mux includes a resistor ladder, which can be used to select a set of gain values. The gain can be selected by the RESSEL bit-field located in DACn_OPAxMUX register. The gain values are taken from tappings of the resistor ladder based on ratio of R2/R1. It is also possible to bypass the resistor ladder in Unity Gain (UG) mode.

Offset Calibration

The offset calibration registers are located in different registers for the opamps. OPA0 and OPA1's offset can be set through the CH0OFFSET and CH1OFFSET bit-fields respectively in DACn_CAL. The offset for OPA2 can be set through OPA2OFFSET in DACn_OPAOFFSET.

Shorting Non-inverting and Inverting Input

Functionality for offset calibration of the opamps has been added, this functionality is enabled by setting the OPAxSHORT bit-field in DACn_OPAxCTRL. Setting this bit-field enables a switch that shorts between the inverting and non-inverting input of the OPA, effectively driving the offset voltage of the opamp to the output. Using

the ADC to measure this offset, the calibration register can be adjusted to minimize the output offset.

Low Pass Filter

The low pass filter is located between the pad and the positive input. The low-pass filter is designed to couple the input signal to local VSS for high frequencies and has a 3 dB frequency of approximately 130 MHz when driven from a 50 ohm source. The filter adds a parasitic capacitance of approximately 1.2 pF towards local VSS when enabled. The filter is enabled out of reset and can be disabled by setting OPAxLPFDIS in DACn_OPAxCTRL.

Disabling of rail-to-rail Operation

Each opamp can have the input rail-to-rail stage disabled by setting the OPAx-HCMDIS bit-field in DACn_OPACTRL. 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.

AD.1.2 Opamp Modes

The opamp can be configured to perform different Operational Amplifier functions by configuring the internal signal routing between the opamps. The modes available are described in the following sections.

General Opamp Mode

In this mode the resistor ladder is isolated from the feedback path and input signal routing is defined by OPAxPOSSEL and OPAxNEGSEL in DACn_OPAxMUX. The output signal routing is defined by OUTPEN in DACn_OPAxMUX

Figure 290: General Opamp Mode Configuration

OPA bit-fields	OPA Configuration
OPAx POSSEL	POSPADx
OPAx NEGSEL	OPATAP, UG, NEGPADx
OPAx RESINMUX	NEXTOUT, POSPADx, NEGPADx VSS

Voltage Follower Unity Gain

In this mode the unity gain feedback path is selected for the negative input by setting the OPAxNEGSEL bit-field to UG in the DACn_OPAxMUX register as shown in Figure 291. The positive input is selected by the OPAxPOSSEL bit-field, and the output is configured by the OUTPEN bit-field, both in the DACn_OPAxMUX register.

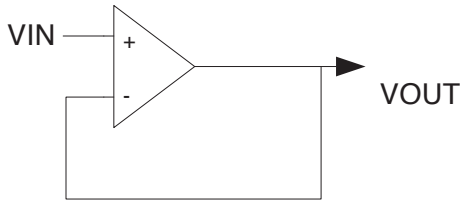


Figure 291: Voltage Follower Unity Gain Overview

OPA bit-fields	OPA Configuration
OPAx POSSEL	OPATAP, NEXTOUT, POSPADx
OPAx NEGSEL	UG
OPAx RESINMUX	DISABLE

Figure 292: Voltage Follower Unity Gain Configuration

Inverting input PGA

Figure 293 shows the inverting input PGA configuration. In this mode the negative input is connected to the resistor ladder by setting the OPAxNEGSEL bit-field to OPATAP in the DACn_OPAxMUX register. This setting provides a programmable gain on the negative input, which can be set by choosing the wanted gain value in the RESSEL bit-field in DACn_OPAxMUX. Signal ground for the positive input can be generated off-chip through the pad by setting OPAxPOSSEL bit-field to PAD in DACn_OPAxMUX. In addition the output is configured by the OUTPEN bit-field, located in DACn_OPAxMUX.

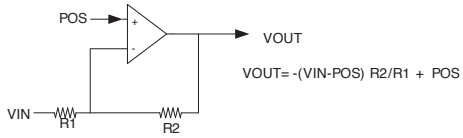


Figure 293: Inverting input PGA Overview

OPA bit-fields	OPA Configuration
OPAx POSSEL	POSPADx
OPAx NEGSEL	OPATAP
OPAx RESINMUX	NEXTOUT, NEGPADx, POSPADx

Figure 294: Inverting input PGA Configuration

Non-inverting PGA

Figure 295 shows the non-inverting input configuration. In this mode the negative input is connected to the resistor ladder by setting the OPAxNEGSEL bit-field to OPATAP in DACn_OPAxMUX. This setting provides a programmable gain on the negative input, which can be set by choosing the wanted gain value in the RESSEL bit-field in DACn_OPAxMUX. In addition the OPAxRESINMUX bit-field must be set to VSS or NEGPAD in DACn_OPAxMUX. The positive input is selected by the

OPAxPOSSEL bit-field, and the output is configured by the OUTPEN bit-field, both located in DACn_OPAxMUX.

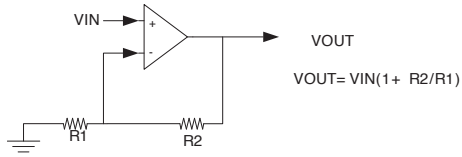


Figure 295: Non-inverting PGA Overview

OPA bit-fields	OPA Configuration
OPAx POSSEL	NEXTOUT, POSPADx
OPAx NEGSEL	OPATAP
OPAx RESINMUX	VSS, NEGPAD

Figure 296: Non-inverting PGA Configuration

Cascaded Inverting PGA

This mode enables the opamp signals to be internally configured to cascade two or three opamps in inverting mode as shown in Figure 297. In both cases the positive input will be configured to signal ground by setting OPAxPOSSEL bit-field to PAD in DACn_OPAx_MUX. When cascaded, the negative input is connected to the resistor ladder by setting the OPAxNEGSEL bit-field to OPATAP in DACn_OPAxMUX. The input to the resistor ladder can be configured in the OPAxRESINMUX bit-field in DAC_nOPAxMUX. The output from OPA0 can be connected to OPA1 to create the second stage by setting the NEXTOUT bit-field in DACn_OPAxMUX. To complete the stage, OPA1RESINMUX field must be set to OPA0INP. Similarly, the last stage can be created by setting the NEXTOUT bit-field in DACn_OPA1MUX and OPA2RESINMUX bit-field to OPA1INP in DACn_OPA2MUX.

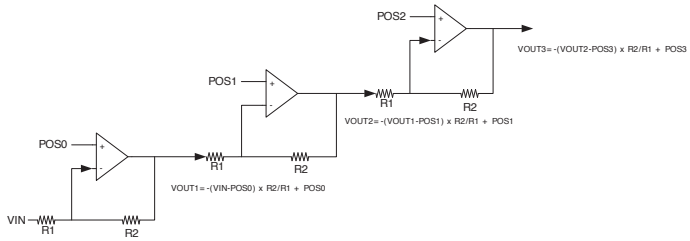


Figure 297: Cascaded Inverting PGA Overview

Cascaded Non-inverting PGA

This mode enables the opamp signals to be internally configured to cascade two or three opamps in non-inverting mode as shown in Figure 299. In both cases the negative input for all opamps will be connected to the resistor ladder by setting the OPAxNEGSEL bit-field to OPATAP. In addition the resistor ladder input must be set to VSS or NEGPADx in the OPAxRESINMUX in DACn_OPAxMUX. When cascaded, the positive input on OPA0 is configured by the OPA0POSSEL bit-field. The output from OPA0 can be connected to OPA1 to create the second stage by setting NEXTOUT

OPA	OPA bit-fields	OPA Configuration
OPA0	POSSEL	POSPAD0
OPA0	NEGSEL	OPA0TAP
OPA0	RESINMUX	NEGPAD0
OPA0	NEXTOUT	1
OPA1	POSSEL	POSPAD1
OPA1	NEGSEL	OPATAP
OPA1	RESINMUX	OPA0INP
OPA1	NEXTOUT	1
OPA2	POSSEL	POSPAD2
OPA2	NEGSEL	OPATAP
OPA2	RESINMUX	OPA1INP

Figure 298: Cascaded Inverting PGA Configuration

in DACn_OPA0MUX. To complete the stage, the OPA1POSSEL bit-field must be set to OPA0INP in DACn_OPA1MUX. Similarly, the last stage can be created by setting NEXTOUT in DACn_OPA1MUX and OPA2POSSEL bit-field to OPA1INP in DACn_OPA2MUX.

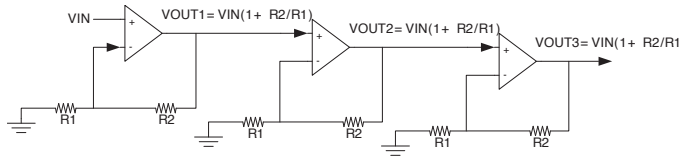


Figure 299: Cascaded Non-inverting PGA Overview

Two Opamp Differential Amplifier

This mode enables OPA0 and OPA1 or OPA1 and OPA2 to be internally configured to form a two opamp differential amplifier as shown in Figure 301. When using OPA0 and OPA1, the positive input of OPA0 can be connected to any input by configuring the OPA0POSSEL bit-field in DACn_OPA0MUX. The OPA0 feedback path must be configured to unity gain by setting the OPA0NEGSEL bit-field to UG in DACn_OPA0MUX. In addition, the OPA0RESINMUX bit-field must be set to DISABLED. The OPA0OUT must be connected to OPA1 by setting NEXTOUT in DACn_OPA0MUX, and OPA1RESINMUX to OPA0INP. The positive input on OPA1 can be set by configuring OPA1POSSEL. The OPA1 output can be configured by configuring the OUTPEN and OUTMODE bit-field.

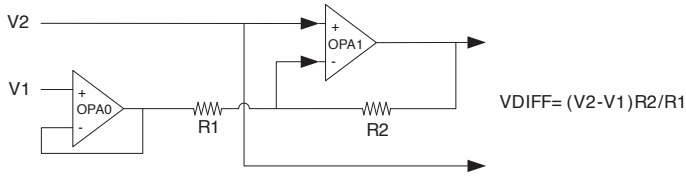
When using OPA1 and OPA2, the positive input of OPA1 can be connected to any input by configuring the OPA1POSSEL bit-field in DACn_OPA1MUX. The OPA1 feedback path must be configured to unity gain by setting the OPA1NEGSEL bit-field to UG in DACn_OPA1MUX. In addition, the OPA1RESINMUX bit-field must be set to DISABLED. The OPA1OUT must be connected to OPA2 by setting NEXTOUT in

OPA	OPA bit-fields	OPA Configuration
OPA0	POSSEL	POSPAD0
OPA0	NEGSEL	OPATAP
OPA0	RESINMUX	VSS, NEGPAD0
OPA0	NEXTOUT	1
OPA1	POSSEL	OPA0INP
OPA1	NEGSEL	OPATAP
OPA1	RESINMUX	VSS, NEGPAD1
OPA1	NEXTOUT	1
OPA2	POSSEL	OPA1INP
OPA2	NEGSEL	OPATAP
OPA2	RESINMUX	VSS, NEGPAD2

Figure 300: Cascaded Non-inverting PGA Configuration

DACn_OPA1MUX, and OPA2RESINMUX to OPA1INP. The positive input on OPA2 can be set by configuring OPA2POSSEL. The OPA2 output can be configured by configuring the OUTPEN and OUTMODE bit-field.

When making a differential connection with the ADC, only OPA1 and OPA2 can be used.



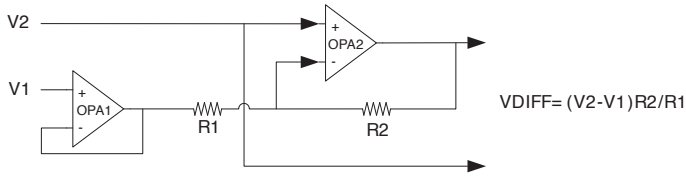


Figure 301: Two Op-amp Differential Amplifier Overview

Three Opamp Differential Amplifier

This mode enables the three opamps to be internally configured to form a three opamp differential amplifier as shown in Figure 304. Both OPA0 and OPA1 can be configured in the same unity gain mode. For both OPA0/OPA1 the positive input can be connected to any input by configuring the OPA0POSSEL/OPA1POSSEL bit-field. The OPA0/OPA1 feedback path must be configured to unity gain by setting the OPA0NEGSEL/OPA1NEGSEL bit-field to UG. In addition the OPA0RESINMUX/OPA1RESINMUX

OPA	OPA bit-fields	OPA Configuration
OPA0	POSSEL	POSPAD1
OPA0	NEGSEL	UG
OPA0	RESINMUX	DISABLE
OPA0	NEXTOUT	1
OPA1	POSSEL	POSPAD1
OPA1	NEGSEL	OPATAP
OPA1	RESINMUX	OPA1INP

Figure 302: OPA0/OPA1 Differential Amplifier Configuration

OPA	OPA bit-fields	OPA Configuration
OPA1	POSSEL	POSPAD1
OPA1	NEGSEL	UG
OPA1	RESINMUX	DISABLE
OPA1	NEXTOUT	1
OPA2	POSSEL	POSPAD1
OPA2	NEGSEL	OPATAP
OPA2	RESINMUX	OPA1INP

Figure 303: OPA1/OPA2 Differential Amplifier Configuration

bit-fields must be set to DISABLED. The OPA1 output must be connected to OPA2 by setting the NEXTOUT bit-field in DACn_OPA1MUX and OPA2RESINMUX to OPA1INP in DACn_OPA2MUX. In addition the OPA2POSSEL must be set to 0PATAP. The OPA2 output can be configured by configuring the OUTPEN and OUTMODE bit-field.

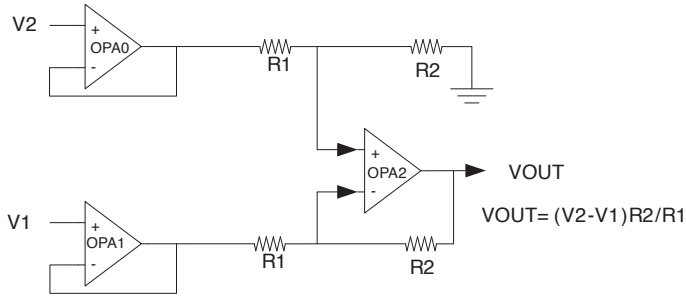


Figure 304: Three Op-amp Differential Amplifier Overview

The gain values for the Three Opamp Differential Amplifier is determined by the combination of the gain settings of OPA0 and OPA2. The 3 different gain values available, 1/3, 1 and 3, can be programmed as shown in the table below.

Dual Buffer ADC Driver

Figure 305: Three Opamp Differential Amplifier Gain Programming

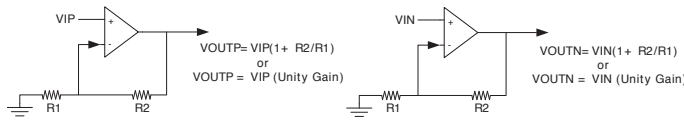
Gain	OPA0 RESSEL	OPA2 RESSEL
1/3	4	0
1	1	1
3	0	4

Figure 306: Three Opamp Differential Amplifier Configuration

OPA	OPA bit-fields	OPA Configuration
OPA0	POSSEL	POSPAD
OPA0	NEGSEL	UG
OPA0	RESINMUX	DISABLE
OPA1	POSSEL	POSPAD
OPA1	NEGSEL	UG
OPA1	RESINMUX	DISABLE
OPA1	NEXTOUT	1
OPA2	POSSEL	OPATAP
OPA2	NEGSEL	OPATAP
OPA2	RESINMUX	OPA1INP

It is possible to use OPA0 and OPA1 to form a Dual Buffer ADC driver as shown in Figure 307. Both opamps used can be configured in the same way. The positive input is configured by setting the 0PAxPOSSEL to PAD and the negative input can be connected to the resistor ladder by setting OPATAP in DACn_OPAxMUX. The output from the opamps can be configured to connect to the ADC by setting OUTMODE to ALT or ALL in DACn_OPAxMUX.

Figure 307: Dual Buffer ADC Driver Overview



AD.1.3 Opamp DAC Combination

Since two of the opamps are part of the DAC it is not possible to use both DAC channels and all three opamps at the same time. If both DAC channels are used, only OPA2 is available out of the 3 opamps. However, it is possible to use one of the DAC channels in combination with OPA0/OPA1. OPA1 is available when DAC channel 0 is in use and OPA0 is available when DAC channel 1 is used. When using

OPA	OPA bit-fields	OPA Configuration
OPA0	POSSEL	POSPAD0
OPA0	NEGSEL	OPATAP
OPA0	RESINMUX	VSS
OPA1	POSSEL	POSPAD1
OPA1	NEGSEL	OPATAP
OPA1	RESINMUX	VSS

Figure 308: Dual Buffer ADC Driver Configuration

the opamp DAC combination, the DAC CONVMODE can only be configured to either CONTINUOUS or SAMPLEHOLD mode. The CONVMODE bitfield can be configured in DACn_CTRL register. In the opamp/DAC combination, the DAC channel enabled is configured through the DAC registers while the opamp is controlled through the opamp registers.

AD.2 Register Description

The register description of the opamp can be found in the DAC chapter.

AD.3 Register Map

The register map of the opamp can be found in the DAC chapter.

AE ARM Advanced Encryption Standard Accelerator

AE.1 Introduction

The Advanced Encryption Standard (FIPS-197) is a symmetric block cipher operating on 128-bit blocks of data and 128-, 192- or 256-bit keys.

The AES accelerator performs AES encryption and decryption with 128-bit or 256-bit keys. Encrypting or decrypting one 128-bit data block takes 54 HFCORECLK cycles with 128-bit keys and 75 HFCORECLK cycles with 256-bit keys. The AES module is an AHB slave which enables efficient access to the data and key registers. All write accesses to the AES module must be 32-bit operations, i.e. 8- or 16-bit operations are not supported.

AE.2 Features

- AES hardware encryption/decryption
  - 128-bit key (54 HFCORECLK cycles)
  - 256-bit key (75 HFCORECLK cycles)
- Efficient CPU/DMA support
- Interrupt on finished encryption/decryption
- DMA request on finished encryption/decryption
- Key buffer in AES128 mode
- Optional XOR on Data write
- Configurable byte ordering

AE.3 Functional Description

Some data and a key must be loaded into the KEY and DATA registers before an encryption or decryption can take place. 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 one encryption, the resulting key in the KEY registers is the CipherKey. This key must be loaded into the KEY registers before every 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 309.

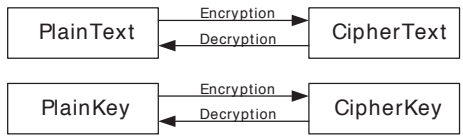


Figure 309: AES Key and Data Definitions

AE.3.1 Encryption/Decryption

The AES module can be set to encrypt or decrypt by clearing/setting the DECRYPT bit in AES_CTRL. The AES256 bit in AES_CTRL configures the size of the key used for encryption/decryption. The AES_CTRL register should not be altered while AES is running, as this may lead to unpredictable behaviour.

An AES encryption/decryption can be started in the following ways:

- Writing a 1 to the START bit in AES_CMD
- Writing 4 times 32 bits to AES_DATA when the DATASTART control bit is set
- Writing 4 times 32 bits to AES_XORDATA when the XORSTART control bit is set

An AES encryption/decryption can be stopped by writing a 1 to the STOP bit in AES_CMD. The RUNNING bit in AES_STATUS indicates that an AES encryption/decryption is ongoing.

AE.3.2 Data and Key Access

The AES module contains a 128-bit DATA (State) register and two 128-bit KEY registers defined as DATA3-DATA0, KEY3-KEY0 (KEYL) and KEY7-KEY4 (KEYH). In AES128 mode, the 128-bit key is read from KEYL, while both KEYH and KEYL are used in AES256 mode. The AES module has configurable byte ordering which is configured in BYTEORDER in AES_CTRL. Figure 310 illustrates how data written to the AES registers is mapped to the key and state defined in the Advanced Encryption Standard (FIPS-197). The figure presents the key byte order for 256-bit keys. In 128-bit mode with BYTEORDER cleared, a_{16} represents the first byte of the 128-bit key. When BYTEORDER is set, a_0 represents the first byte in the key. AES encryption/decryption takes two extra cycles when BYTEORDER is set. BYTEORDER has to be set prior to loading the data and key registers.

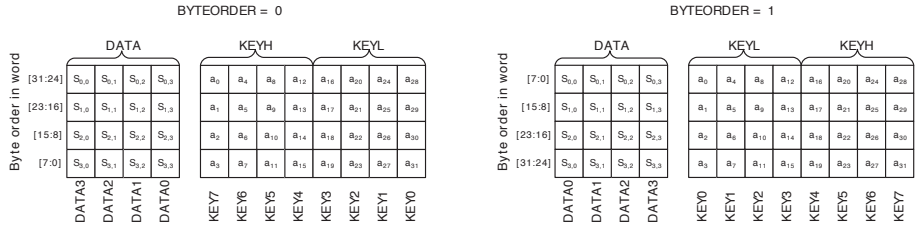


Figure 310: AES Data and Key Orientation as Defined in the Advanced Encryption Standard

The registers DATA3-DATA0, are not memory mapped directly, but can be written/read by accessing AES_DATA or AES_XORDATA. The same applies for the key registers, KEY3-KEY0 which are accessed through AES_KEYLn (n=A, B, C or D), while KEY7-KEY4 are accessed through KEYHn (n=A, B, C or D). Writing DATA3-DATA0 is then done through 4 consecutive writes to AES_DATA (or AES_XORDATA), starting with the word which is to be written to DATA0. For each write, the words will be

word wise barrel shifted towards the least significant word. Accessing the KEY registers are done in the same fashion through KEYLn and KEYHn. See Figure 311. Note that KEYHA, KEYHB, KEYHC and KEYHD are really the same register, just mapped to four different addresses. You can then choose freely which of these addresses you want to use to update the KEY7-KEY4 registers. The same principle applies to the KEYLn registers. Mapping the same registers to multiple addresses like this, allows the DMA controller to write a full 256-bit key in one sweep, when incrementing the address between each word write.

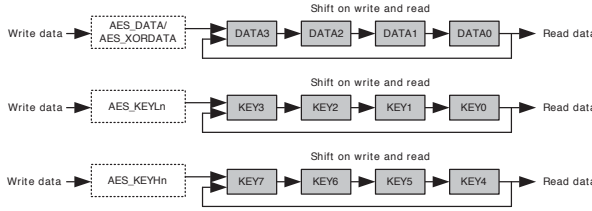


Figure 311: AES Data and Key Register Operation

Key Buffer

When encrypting multiple blocks of data in a row, the PlainKey must be written to the key register between each encryption, since the contents of the key registers will be turned into the CipherKey during the encryption. The opposite applies when decrypting, where you have to re-supply the CipherKey between each block. However, in AES128 mode, KEY4-KEY7 can be used as a buffer register, to hold an extra copy of the KEY3-KEY0 registers. When KEYBUFEN is set in AES_CTRL, the contents of KEY7-KEY4 are copied to KEY3-KEY0, when an encryption/decryption is started. This eliminates the need for re-loading the KEY for every encrypted/decrypted block when running in AES128 mode.

Data Write XOR

The AES module contains an array of XOR gates connected to the DATA registers, which can be used during a data write to XOR the existing contents of the registers with the new data written. To use the XOR function, the data must be written to AES_XORDATA location.

Reading data from AES_XORDATA is equivalent to reading data from AES_DATA.

Start on Data Write

The AES module can be configured to start an encryption/decryption when the new data has been written to AES_DATA and/or AES_XORDATA. A 2-bit counter is incremented each time the AES_DATA or AES_XORDATA registers are written. This counter indicates which data word is written. If DATASTART/XORSTART in AES_CTRL is set, an encryption will start each time the counter overflows (DATA3 is written). Writing to the AES_CTRL register will reset the counter to 0.

AE.3.3 Interrupt Request

The DONE interrupt flag is set when an encryption/ decryption has finished.

AE.3.4 DMA Request

The AES module has 4 DMA requests which are all set on a finished encryption/decryption and cleared on the following conditions:

- DATAWR: Cleared on a AES_DATA write or AES_CTRL write
- XORDATAWR: Cleared on a AES_XORDATA write or AES_CTRL write
- DATARD: Cleared on a AES_DATA read or AES_CTRL write
- KEYWR: Cleared on a AES_KEYHn write or AES_CTRL write

AE.3.5 Block Chaining Example

Example AE.3.5 below illustrates how the AES module could be configured to perform Cipher Block Chaining with 128-bit keys.

exampleAES Cipher Block Chaining

1. Configure module to encryption, key buffer enabled and XORSTART in AES_CTRL.
2. Write 128-bit initialization vector to AES_DATA, starting with least significant word.
3. Write PlainKey to AES_KEYHn, starting with least significant word.
4. Write PlainText to AES_XORDATA, starting with least significant word. Encryption will be started when the DATA3 is written. KEYH (PlainKey) will be copied to KEYL before encryption starts.
5. When encryption finished, read CipherText from AES_DATA, starting with least significant word.
6. Loop to step 4, if new PlainText is available.

AE.4 Register Map

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

Offset	Name	Type	Description
0x000	AES_CTRL	RW	Control Register
0x004	AES_CMD	W1	Command Register

Offset	Name	Type	Description
0x008	AES_STATUS	R	Status Register
0x00C	AES_IEN	RW	Interrupt Enable Register
0x010	AES_IF	R	Interrupt Flag Register
0x014	AES_IFS	W1	Interrupt Flag Set Register
0x018	AES_IFC	W1	Interrupt Flag Clear Register
0x01C	AES_DATA	RW	DATA Register
0x020	AES_XORDATA	RW	XORDATA Register
0x030	AES_KEYLA	RW	KEY Low Register
0x034	AES_KEYLB	RW	KEY Low Register
0x038	AES_KEYLC	RW	KEY Low Register
0x03C	AES_KEYLD	RW	KEY Low Register
0x040	AES_KEYHA	RW	KEY High Register
0x044	AES_KEYHB	RW	KEY High Register
0x048	AES_KEYHC	RW	KEY High Register
0x04C	AES_KEYHD	RW	KEY High Register

AE.5 Register Description

AE.5.1 AES_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																										BYTEORDER	XORSTART	DATASTART		KEYBUFEN	AES256	DECRYPT

Bit	Name	Reset	Access	Description
31:7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6	BYTEORDER	0	RW	**Configure byte order in data and key registers**
	When set, the byte orders in the data and key registers are swapped before and after encryption/decryption.			
5	XORSTART	0	RW	**AES_XORDATA Write Start**
	Set this bit to start encryption/decryption when DATA3 is written through AES_XORDATA.			
4	DATASTART	0	RW	**AES_DATA Write Start**
	Set this bit to start encryption/decryption when DATA3 is written through AES_DATA.			
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	KEYBUFEN	0	RW	**Key Buffer Enable**
	Enable/disable key buffer in AES-128 mode.			
1	AES256	0	RW	**AES-256 Mode**

Bit	Name	Reset	Access	Description
	Select AES-128 or AES-256 mode.			

Value	Description
0	AES-128 mode
1	AES-256 mode

Bit	Name	Reset	Access	Description
0	DECRYPT	0	RW	**Decryption/Encryption Mode**
	Select encryption or decryption.			

Value	Description
0	AES Encryption
1	AES Decryption

AE.5.2 AES_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.		
1	STOP	0	W1	**Encryption/Decryption Stop**
	Set to stop encryption/decryption.			
0	START	0	W1	**Encryption/Decryption Start**
	Set to start encryption/decryption.			

AE.5.3 AES_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.		
0	RUNNING	0	R	**AES Running**
	This bit indicates that the AES module is running an encryption/decryption.			

AE.5.4 AES_IEN - Interrupt Enable 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

Offset	Bit Position	
Access		RW
Name		DONE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	DONE	0	RW	**Encryption/Decryption Done Interrupt Enable**
	Enable/disable interrupt on encryption/decryption done.			

AE.5.5 AES_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																																DONE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	DONE	0	R	**Encryption/Decryption Done Interrupt Flag**
	Set when an encryption/decryption has finished.			

AE.5.6 AES_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																																DONE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	DONE	0	W1	**Encryption/Decryption Done Interrupt Flag Set**
	Write to 1 to set encryption/decryption done interrupt flag			

AE.5.7 AES_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																																W1
Name																																DONE

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	DONE	0	W1	**Encryption/Decryption Done Interrupt Flag Clear**
	Write to 1 to clear encryption/decryption done interrupt flag			

AE.5.8 AES_DATA - 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	RW																															
Name	DATA																															

Bit	Name	Reset	Access	Description
31:0	DATA	0x00000000	RW	**Data Access**
	Access data through this register.			

AE.5.9 AES_XORDATA - XORDATA 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	XORDATA																															

Bit	Name	Reset	Access	Description
31:0	XORDATA	0x00000000	RW	**XOR Data Access**
	Access data with XOR function through this register.			

AE.5.10 AES_KEYLA - KEY Low 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	RW																															
Name	KEYLA																															

Bit	Name	Reset	Access	Description
31:0	KEYLA	0x00000000	RW	**Key Low Access A**
	Access the low key words through this register.			

AE.5.11 AES_KEYLB - KEY Low 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	RW																															
Name	KEYLB																															

Bit	Name	Reset	Access	Description
31:0	KEYLB	0x00000000	RW	**Key Low Access B**
	Access the low key words through this register.			

AE.5.12 AES_KEYLC - KEY Low 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	RW																															
Name	KEYLC																															

Bit	Name	Reset	Access	Description
31:0	KEYLC	0x00000000	RW	**Key Low Access C**
	Access the low key words through this register.			

AE.5.13 AES_KEYLD - KEY Low 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	0x00000000																															
Access	RW																															
Name	KEYLD																															

Bit	Name	Reset	Access	Description
31:0	KEYLD	0x00000000	RW	**Key Low Access D**
	Access the low key words through this register.			

AE.5.14 AES_KEYHA - KEY High 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	0x00000000																															
Access	RW																															
Name	KEYHA																															

Bit	Name	Reset	Access	Description
31:0	KEYHA	0x00000000	RW	**Key High Access A**

Bit	Name	Reset	Access	Description
	Access the high key words through this register.			

AE.5.15 AES_KEYHB - KEY High 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	0x00000000																															
Access	RW																															
Name	KEYHB																															

Bit	Name	Reset	Access	Description
31:0	KEYHB	0x00000000	RW	**Key High Access B**
	Access the high key words through this register.			

AE.5.16 AES_KEYHC - KEY High 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	0x00000000																															
Access	RW																															
Name	KEYHC																															

Bit	Name	Reset	Access	Description
31:0	KEYHC	0x00000000	RW	**Key High Access C**
	Access the high key words through this register.			

AE.5.17 AES_KEYHD - KEY High 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	0x00000000																															
Access	RW																															
Name	KEYHD																															

Bit	Name	Reset	Access	Description
31:0	KEYHD	0x00000000	RW	**Key High Access D**
	Access the high key words through this register.			

AF ARM General Purpose Input/Output

AF.1 Introduction

In the ARM core the General Purpose Input/Output (GPIO) pins are organized into ports with up to 16 pins each. These pins can individually be configured as either an output or input. More advanced configurations like open-drain, filtering and drive strength can also be configured individually for the pins. 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 enables 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.

AF.2 Features

- Individual configuration for each pin
  - Tristate (reset state)
  - Push-pull
  - Open-drain
  - Pull-up resistor
  - Pull-down resistor
  - Drive strength
    - 0.5 mA
    - 2 mA
    - 6 mA
    - 20 mA
- EM4 IO pin retention. This includes
  - Output enable
  - Output value
  - Pull enable
  - Pull direction
- EM4 wake-up on selected GPIO pins
- Glitch suppression input filter.
- Analog connection to e.g. ADC.
- 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, set and clear registers for output data
- Dedicated data input register (read-only)
- Interrupts
  - 2 interrupt lines from up to 16 pending sources
    - All GPIO pins are selectable
  - Separate enable, status, set and clear registers
  - Asynchronous sensing
  - Rising, falling or both edges
  - Wake up from EM0-EM3
- Peripheral Reflex System producer
  - All GPIO pins are selectable
- Configuration lock functionality to avoid accidental changes

AF.3 Functional Description

An overview of the GPIO module is shown in Figure 312.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 is disabled for all pins on the device, except for debug pins. To use a pin, the port GPIO_Px_MODEL/GPIO_Px_MODEH registers 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 Section AF.3.1. When the port is either 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 a 1 to a bit in the GPIO_Px_DOUTSET register sets the corresponding DOUT bit
- Writing a 1 to a bit in the GPIO_Px_DOUTCLR register clears the corresponding DOUT bit
- Writing a 1 to a bit in the GPIO_Px_DOUTTGL register toggles the corresponding DOUT bit

Reading the GPIO_Px_DOUT register will return its contents. Reading the GPIO_Px_DOUTSET, GPIO_Px_CLR or GPIO_Px_TGL will return 0.

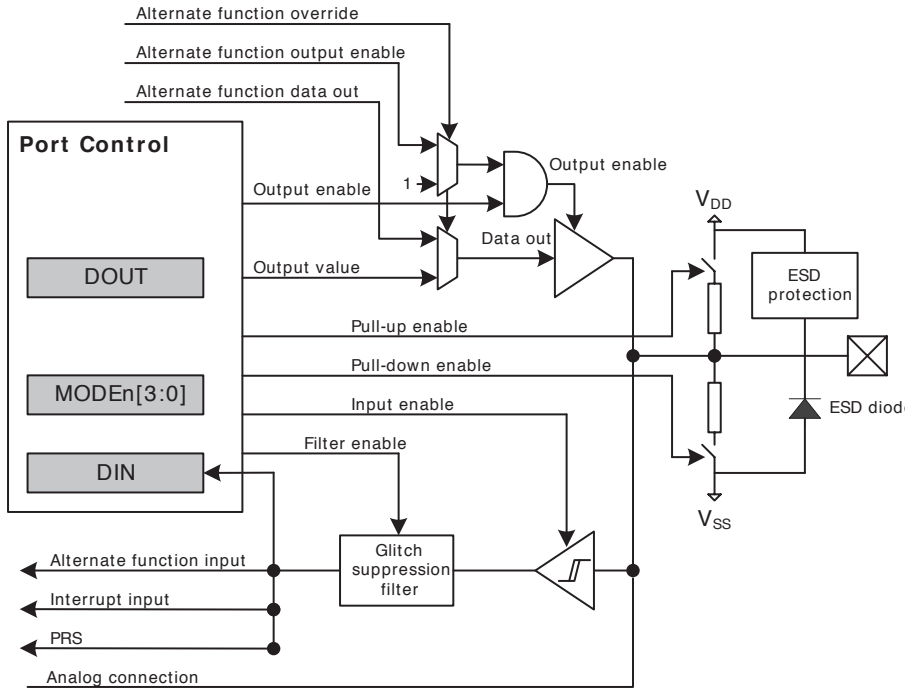


Figure 312: Pin Configuration

There is no ESD diode to Vdd. Nevertheless there is an ESD protection block against over voltage.

AF.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 313 shows the available configurations.

MODEn determines which mode the pin is in at a given time. Setting MODEn to 0b0000 disables the pin, reducing power consumption to a minimum. When the output driver is disabled, the pin can be used as a connection for an analog module (e.g. ADC). Input is enabled by setting MODEn to any value other than 0b0000. The pull-up, pull-down and filter function can optionally be applied to the input, see Figure 314.

The internal pull-up resistance, R_{PU}, and pull-down resistance, R_{PD}, are defined in the device datasheet. When the filter is enabled it suppresses glitches with pulse widths as defined by the parameter $t_{IOGLITCH}$ in the device datasheet.

MODEn	Input	Output	DOUT	Pull-down	Pull-up	Alt. strength	Input Filter	Description
0b0000	Disabled	Disabled	0					Input disabled
			1		On			Input disabled with pull-up
0b0001	Enabled		0					Input enabled
			1				On	Input enabled with filter
0b0010			0	On				Input enabled with pull-down
			1		On			Input enabled with pull-up
0b0011			0	On			On	Input enabled with pull-down and filter
			1		On		On	Input enabled with pull-up and filter
0b0100		Push-pull	x					Push-pull
0b0101			x			On		Push-pull with alt. drive strength
0b0110	Enabled	Open Source (Wired-OR)	x					Open-source
0b0111			x	On				Open-source with pull-down
0b1000	Enabled	Open Drain (Wired-AND)	x					Open-drain
0b1001			x				On	Open-drain with filter
0b1010			x		On			Open-drain with pull-up
0b1011			x		On		On	Open-drain with pull-up and filter
0b1100			x			On		Open-drain with alt. drive strength
0b1101			x			On	On	Open-drain with alt. drive strength and filter
0b1110			x		On	On		Open-drain with alt. drive strength and pull-up
0b1111			x		On	On	On	Open-drain with alt. drive strength, pull-up and filter

Figure 313: Pin Configuration

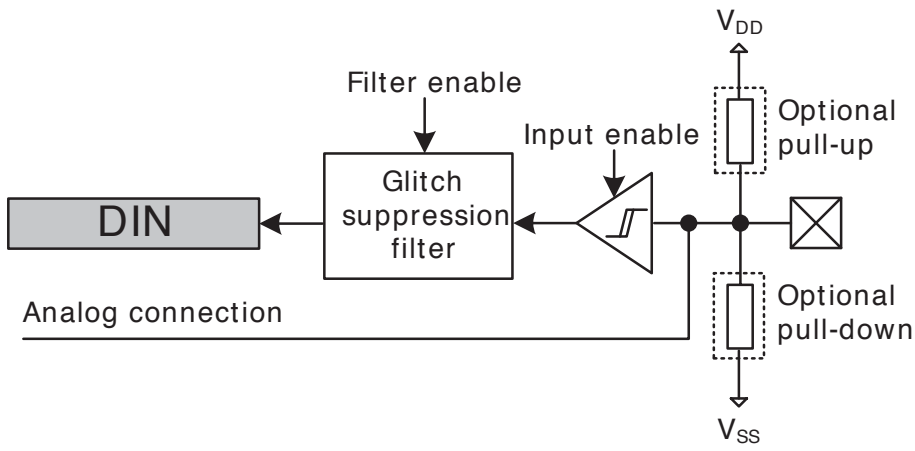


Figure 314: Tristated Output with Optional Pull-up or Pull-down

When MODEn=0b0100 or MODEn=0b0101, the pin operates in push-pull mode. In this mode, the pin is driven either high or low, dependent on the value of GPIO_Px_DOUT. The push-pull configuration is shown in Figure 315.

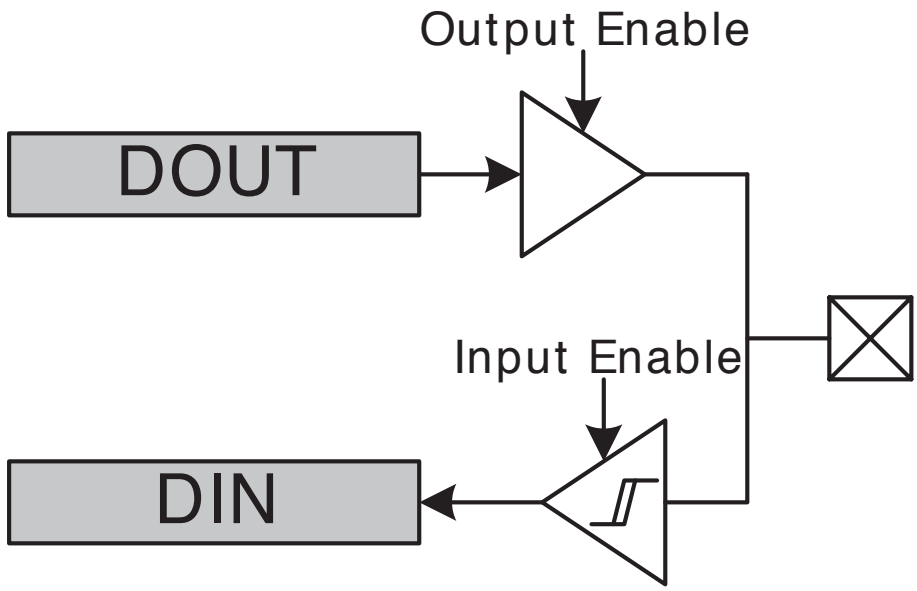


Figure 315: Push-Pull Configuration

When MODEn is 0110 or 0111, the pin operates in open-source mode, the latter with a pull-down resistor. When driving a high value in open-source mode, the pull-down is disconnected to save power.

For the remaining MODEn values, i.e. MODEn >= 1000, the pin operates in open-drain mode as shown in Figure 316. In open-drain mode, the pin can have an input filter, a pull-up, different driver strengths or any combination of these. When driving a low value in open-drain mode, the pull-up is disconnected to save power.

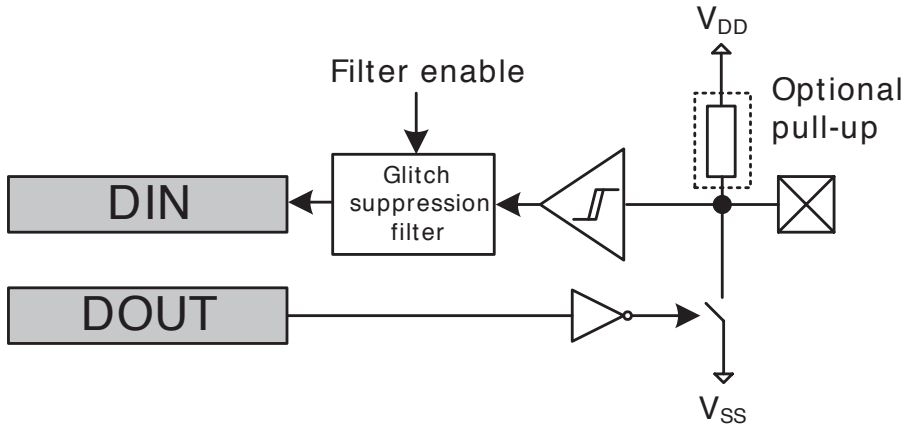


Figure 316: Open-drain

When MODEn=0b0101 or 0b11xx, the output driver uses the drive strength specified in DRIVEMODE in GPIO_Px_CTRL. In all other output modes, the drive strength is set to 6 mA.

Configuration Lock

GPIO_Px_MODEL, GPIO_Px_MODEH, GPIO_Px_CTRL, GPIO_Px_PINLOCKN, GPIO_EXTIPSELL, GPIO_EXTIPSELH, GPIO_INSENSE and GPIO_ROUTE can be locked by writing any other value 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_DOUTSET, GPIO_Px_DOUTCLR, and GPIO_Px_DOUTTGL can be locked individually for each pin by clearing the corresponding bit in GPIO_Px_PINLOCKN. Bits in the GPIO_Px_PINLOCKN register can only be cleared, they are set high again after reset.

AF.3.2 EM4 Wake-up

It is possible to wake-up from EM4 through reset triggered from any of up to 6 selectable GPIO pins. For the wake-up logic to work correctly, EM4 retention needs to be enabled before entering EM4, as described in Section AF.3.3 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_EM4WUPOL register.

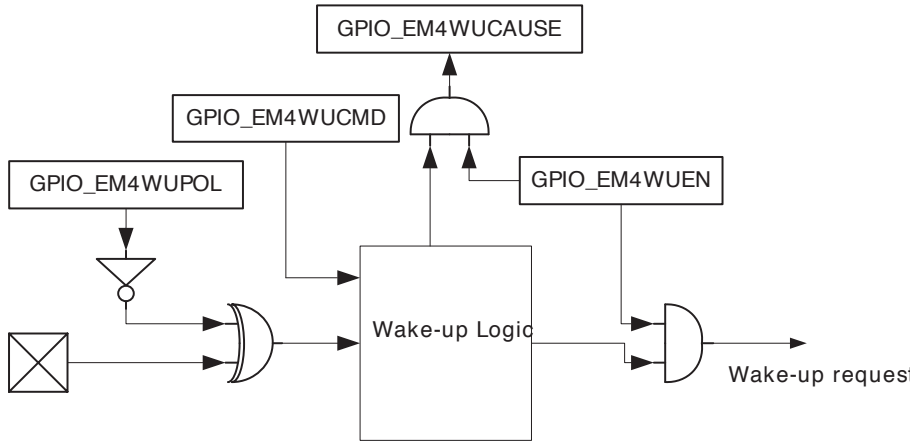


Figure 317: EM4 Wake-up Logic

The pins used for EM4 wake-up must be configured as inputs using the GPIO_Px_MODEL/GPIO_Px register. Before going down to EM4, it is important to clear the wake-up logic by setting the EM4WUCLR bitfield in the GPIO_CMD register, which clears the complete wake-up logic, including the GPIO_EM4WUCAUSE register. When the chip comes

out of reset, it is possible to determine what caused the reset by reading the RMU_RSTCAUSE register. If an EM4 wake-up reset occurred, the EM4RST (indicating the chip was in EM4) and the EM4WU (indicating the EM4 wake-up reset) bits should be set. It is possible to determine which pin caused the reset by reading the GPIO_EM4WUCAUSE register. The mapping between pins and the bits in the GPIO_EM4WUEN, GPIO_EM4WUPOL, and GPIO_EM4WUCAUSE registers are described in Table 318

Figure 318: EM4 WU Register bits to pin mapping

Wake-up Registers Bits	Pin
bit 0	A0
bit 1	A6
bit 2	C9
bit 3	F1
bit 4	F2
bit 5	E13

AF.3.3 EM4 Retention

It is possible to enable retention of output enable, output value, pull enable and pull direction when in EM4. EM4 retention also makes it possible to wake up from EM4 on pin reset as described in Section AF.3.2 EM4 retention can be enabled by setting the EM4RET field in GPIO_CTRL register before going down in EM4.

AF.3.4 Alternate Functions

Alternate functions are connections to pins from Timers, USARTs etc. These modules contain route registers, where the pin connections are enabled. In addition, these registers contain a location bit field, which configures which pins the outputs of that module will be connected to if they are enabled. If an alternate signal output is enabled for a pin and output is enabled for the pin, the alternate function's output data and output enable signals override the data output and output enable signals from the GPIO. However, the pin configuration stays as set in GPIO_Px_MODEL, GPIO_Px_MODEH and GPIO_Px_DOUT registers. I.e. the pin configuration must be set to output enable in GPIO for a peripheral to be able to use the pin as an output.

It is possible, but not recommended to select two or more peripherals as output on the same pin. These signals will then be OR'ed together. However, TIMER CCx and CDTIx outputs, which are routed as alternate functions, have priority, and will never be OR'ed with other alternate functions. The reader is referred to the pin map section of the device datasheet for more information on the possible locations of each alternate function and any priority settings.

Serial Wire Debug Port 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 SWDIOPEN and SWCLKPEN bits in GPIO_ROUTE to 0.

WARNING: 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 default state as enabled. 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 halt the device after a reset before the pins are disabled.

The Serial Wire Viewer Output pin (SWO) can be enabled by setting the SWOPEN bit in GPIO_ROUTE. This bit can also be routed to alternate locations by configuring the LOCATION bitfield in GPIO_ROUTE.

ETM Trace Ports

There are five trace pins available on the device. One trace clock which can be enabled by setting the TCLKPEN bitfield in GPIO_ROUTE. The four data pins can be enabled individually by setting TD0PEN, TD1PEN, TD2PEN, and TD3PEN respectively in GPIO_ROUTE. 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 ETMLOCATION bitifeld in GPIO_ROUTE.

Analog Connections

When using the GPIO pin for analog functionality, it is recommended to disable the digital output and set the MODEn in GPIO_Px_MODEL/GPIO_Px_MODEH equal to 0b0000 to disable the input sense and pull resistors.

AF.3.5 Interrupt Generation

The GPIO can generate an interrupt from the input of any GPIO pin on a device. The interrupts have asynchronous sense capability, enabling wake-up from energy modes as low as EM3, see Figure 319.

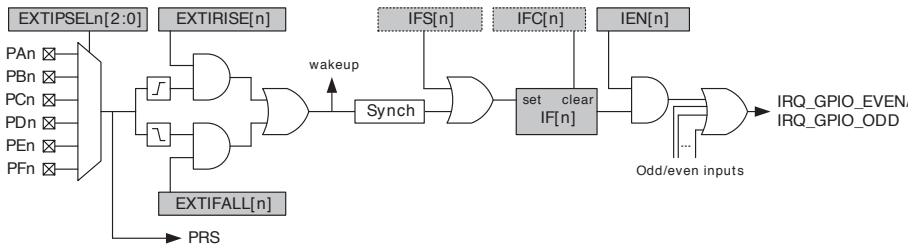


Figure 319: Pin n Interrupt Generation

All pins with the same pin number (n) are grouped together to trigger one interrupt flag (EXT[n] in GPIO_IF). The EXTIPSELn[2:0] bits in GPIO_EXTIPSELL or GPIO_EXTIPSELH select which port will trigger the interrupt flag. The GPIO_EXTIRISE[n] and GPIO_EXTIFALL[n] registers enables 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, while the odd is triggered by odd flags. The interrupt flags can be set and cleared by software by writing the GPIO_IFS and GPIO_IFC registers, see Example AF.3.5. 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 filtering for pins that have external interrupts enabled.

exampleGPIO Interrupt Example

Setting EXTIPSEL3 in GPIO_EXTIPSELL to 2 (Port C) and setting the GPIO_EXTIRISE[3] bit, the interrupt flag EXT[3] in GPIO_IF will be triggered by a rising edge on pin 3 on PORT C. If EXT[3] in GPIO_IEN is set as well, a interrupt request will be sent on IRQ_GPIO_ODD.

AF.3.6 Output to PRS

All pins with the same pin number (n) are grouped together to form one PRS producer output, giving a total of 16 outputs to the PRS. The port on which the output n should be taken is selected by the EXTIPSELn[3:0] bits in the GPIO_EXTIPSELL or the GPIO_EXTIPSELH registers.

AF.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 HFCORECLK. Consequently, when a pin changes state, the change will have propagated to GPIO_Px_DIN after 2 positive HFCORECLK edges, or maximum 2 HFCORECLK cycles.

Synchronization (also running on the HFCORECLK) is also added for interrupt input. The input to the PRS generation is also synchronized, but these flip-flops run on the HFPERCLK. To save power when the external interrupts or PRS generation is not used, the synchronization flip-flops for these can be turned off by clearing the INTSENSE or PRSSENSE, respectively, in GPIO_INSENSE register.

To use the GPIO, the GPIO clock must first be enabled in CMU_HFPERCLKEN0. Setting this bit enables the HFCORECLK and the HFPERCLK for the GPIO. HFCORECLK is used for updating registers, while HFPERCLK is only used to synchronize PRS and interrupts. The PRS and interrupt synchronization can also be disabled through GPIO_INSENSE, if these are not used.

AF.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
0x010	GPIO_PA_DOUTSET	W1	Port Data Out Set Register
0x014	GPIO_PA_DOUTCLR	W1	Port Data Out Clear 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
0x024	GPIO_PB_CTRL	RW	Port Control Register
0x028	GPIO_PB_MODEL	RW	Port Pin Mode Low Register
0x02C	GPIO_PB_MODEH	RW	Port Pin Mode High Register
0x030	GPIO_PB_DOUT	RW	Port Data Out Register
0x034	GPIO_PB_DOUTSET	W1	Port Data Out Set Register
0x038	GPIO_PB_DOUTCLR	W1	Port Data Out Clear Register
0x03C	GPIO_PB_DOUTTGL	W1	Port Data Out Toggle Register
0x040	GPIO_PB_DIN	R	Port Data In Register
0x044	GPIO_PB_PINLOCKN	RW	Port Unlocked Pins Register
0x048	GPIO_PC_CTRL	RW	Port Control Register
0x04C	GPIO_PC_MODEL	RW	Port Pin Mode Low Register
0x050	GPIO_PC_MODEH	RW	Port Pin Mode High Register
0x054	GPIO_PC_DOUT	RW	Port Data Out Register
0x058	GPIO_PC_DOUTSET	W1	Port Data Out Set Register
0x05C	GPIO_PC_DOUTCLR	W1	Port Data Out Clear Register
0x060	GPIO_PC_DOUTTGL	W1	Port Data Out Toggle Register
0x064	GPIO_PC_DIN	R	Port Data In Register
0x068	GPIO_PC_PINLOCKN	RW	Port Unlocked Pins Register
0x06C	GPIO_PD_CTRL	RW	Port Control Register
0x070	GPIO_PD_MODEL	RW	Port Pin Mode Low Register
0x074	GPIO_PD_MODEH	RW	Port Pin Mode High Register

Offset	Name	Type	Description
0x078	GPIO_PD_DOUT	RW	Port Data Out Register
0x07C	GPIO_PD_DOUTSET	W1	Port Data Out Set Register
0x080	GPIO_PD_DOUTCLR	W1	Port Data Out Clear Register
0x084	GPIO_PD_DOUTTGL	W1	Port Data Out Toggle Register
0x088	GPIO_PD_DIN	R	Port Data In Register
0x08C	GPIO_PD_PINLOCKN	RW	Port Unlocked Pins Register
0x090	GPIO_PE_CTRL	RW	Port Control Register
0x094	GPIO_PE_MODEL	RW	Port Pin Mode Low Register
0x098	GPIO_PE_MODEH	RW	Port Pin Mode High Register
0x09C	GPIO_PE_DOUT	RW	Port Data Out Register
0x0A0	GPIO_PE_DOUTSET	W1	Port Data Out Set Register
0x0A4	GPIO_PE_DOUTCLR	W1	Port Data Out Clear Register
0x0A8	GPIO_PE_DOUTTGL	W1	Port Data Out Toggle Register
0x0AC	GPIO_PE_DIN	R	Port Data In Register
0x0B0	GPIO_PE_PINLOCKN	RW	Port Unlocked Pins Register
0x0B4	GPIO_PF_CTRL	RW	Port Control Register
0x0B8	GPIO_PF_MODEL	RW	Port Pin Mode Low Register
0x0BC	GPIO_PF_MODEH	RW	Port Pin Mode High Register
0x0C0	GPIO_PF_DOUT	RW	Port Data Out Register
0x0C4	GPIO_PF_DOUTSET	W1	Port Data Out Set Register
0x0C8	GPIO_PF_DOUTCLR	W1	Port Data Out Clear Register
0x0CC	GPIO_PF_DOUTTGL	W1	Port Data Out Toggle Register
0x0D0	GPIO_PF_DIN	R	Port Data In Register
0x0D4	GPIO_PF_PINLOCKN	RW	Port Unlocked Pins Register
0x100	GPIO_EXTIPSELL	RW	External Interrupt Port Select Low Register
0x104	GPIO_EXTIPSELH	RW	External Interrupt Port Select High Register
0x108	GPIO_EXTIRISE	RW	External Interrupt Rising Edge Trigger Register
0x10C	GPIO_EXTIFALL	RW	External Interrupt Falling Edge Trigger Register
0x110	GPIO_IEN	RW	Interrupt Enable Register
0x114	GPIO_IF	R	Interrupt Flag Register

Offset	Name	Type	Description
0x118	GPIO_IFS	W1	Interrupt Flag Set Register
0x11C	GPIO_IFC	W1	Interrupt Flag Clear Register
0x120	GPIO_ROUTE	RW	I/O Routing Register
0x124	GPIO_INSENSE	RW	Input Sense Register
0x128	GPIO_LOCK	RW	Configuration Lock Register
0x12C	GPIO_CTRL	RW	GPIO Control Register
0x130	GPIO_CMD	W1	GPIO Command Register
0x134	GPIO_EM4WUEN	RW	EM4 Wake-up Enable Register
0x138	GPIO_EM4WUPOL	RW	EM4 Wake-up Polarity Register
0x13C	GPIO_EM4WUCAUSE	R	EM4 Wake-up Cause Register

AF.5 Register Description

AF.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																															0x0	
Access																															RW	
Name																															DRIVEMODE	

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1:0	DRIVEMODE	0x0	RW	**Drive Mode Select**

Select drive mode for all pins on port configured with alternate drive strength.

Value	Mode	Description
0	STANDARD	6 mA drive current
1	LOWEST	0.5 mA drive current
2	HIGH	20 mA drive current
3	LOW	2 mA drive current

AF.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. Enumeration is equal to MODE0.			
27:24	MODE6	0x0	RW	**Pin 6 Mode**
	Configure mode for pin 6. Enumeration is equal to MODE0.			
23:20	MODE5	0x0	RW	**Pin 5 Mode**
	Configure mode for pin 5. Enumeration is equal to MODE0.			
19:16	MODE4	0x0	RW	**Pin 4 Mode**
	Configure mode for pin 4. Enumeration is equal to MODE0.			
15:12	MODE3	0x0	RW	**Pin 3 Mode**
	Configure mode for pin 3. Enumeration is equal to MODE0.			
11:8	MODE2	0x0	RW	**Pin 2 Mode**
	Configure mode for pin 2. Enumeration is equal to MODE0.			
7:4	MODE1	0x0	RW	**Pin 1 Mode**
	Configure mode for pin 1. Enumeration is equal to MODE0.			
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	PUSHPULLDRIVE	Push-pull output with drive-strength set by DRIVEMODE
6	WIREDOR	Wired-or output
7	WIREDORPULLDOWN	Wired-or output with pull-down
8	WIREDAND	Open-drain output
9	WIREDANDFILTER	Open-drain output with filter
10	WIREDANDPULLUP	Open-drain output with pullup
11	WIREDANDPULLUPFILTER	Open-drain output with filter and pullup
12	WIREDANDDRIVE	Open-drain output with drive-strength set by DRIVEMODE
13	WIREDANDDRIVEFILTER	Open-drain output with filter and drive-strength set by DRIVEMODE
14	WIREDANDDRIVEPULLUP	Open-drain output with pullup and drive-strength set by DRIVEMODE
15	WIREDANDDRIVEPULLUPFILTER	Open-drain output with filter, pullup and drive-strength set by DRIVEMODE

AF.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. Enumeration is equal to MODE8.			
27:24	MODE14	0x0	RW	**Pin 14 Mode**
	Configure mode for pin 14. Enumeration is equal to MODE8.			
23:20	MODE13	0x0	RW	**Pin 13 Mode**
	Configure mode for pin 13. Enumeration is equal to MODE8.			
19:16	MODE12	0x0	RW	**Pin 12 Mode**

Bit	Name	Reset	Access	Description
	Configure mode for pin 12. Enumeration is equal to MODE8.			
15:12	MODE11	0x0	RW	**Pin 11 Mode**
	Configure mode for pin 11. Enumeration is equal to MODE8.			
11:8	MODE10	0x0	RW	**Pin 10 Mode**
	Configure mode for pin 10. Enumeration is equal to MODE8.			
7:4	MODE9	0x0	RW	**Pin 9 Mode**
	Configure mode for pin 9. Enumeration is equal to MODE8.			
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	PUSHPULLDRIVE	Push-pull output with drive-strength set by DRIVEMODE
6	WIREDOR	Wired-or output
7	WIREDORPULLDOWN	Wired-or output with pull-down
8	WIREDAND	Open-drain output
9	WIREDANDFILTER	Open-drain output with filter
10	WIREDANDPULLUP	Open-drain output with pullup
11	WIREDANDPULLUPFILTER	Open-drain output with filter and pullup
12	WIREDANDDRIVE	Open-drain output with drive-strength set by DRIVEMODE
13	WIREDANDDRIVEFILTER	Open-drain output with filter and drive-strength set by DRIVEMODE
14	WIREDANDDRIVEPULLUP	Open-drain output with pullup and drive-strength set by DRIVEMODE
15	WIREDANDDRIVEPULLUPFILTER	Open-drain output with filter, pullup and drive-strength set by DRIVEMODE

AF.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.		
15:0	DOUT	0x0000	RW	**Data Out**
	Data output on port.			

AF.5.5 GPIO_Px_DOUTSET - Port Data Out 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																	0x0000															
Access																	W1															
Name																	DOUTSET															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	DOUTSET	0x0000	W1	**Data Out Set**
	Write bits to 1 to set corresponding bits in GPIO_Px_DOUT. Bits written to 0 will have no effect.			

AF.5.6 GPIO_Px_DOUTCLR - Port Data Out 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	0x0000																															
Access	W1																															
Name	DOUTCLR																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	DOUTCLR	0x0000	W1	**Data Out Clear**
	Write bits to 1 to clear corresponding bits in GPIO_Px_DOUT. Bits written to 0 will have no effect.			

AF.5.7 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.		
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.			

AF.5.8 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.		
15:0	DIN	0x0000	R	**Data In**
	Port data input.			

AF.5.9 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.		
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.			

AF.5.10 GPIO_EXTIPSELL - External Interrupt Port Select Low 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				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	Reserved	To ensure compatibility with future devices, always write bits to 0.		
30:28	EXTIPSEL7	0x0	RW	**External Interrupt 7 Port Select**
	Select input port for external interrupt 7.			

Value	Mode	Description
0	PORTA	Port A pin 7 selected for external interrupt 7
1	PORTB	Port B pin 7 selected for external interrupt 7
2	PORTC	Port C pin 7 selected for external interrupt 7
3	PORTD	Port D pin 7 selected for external interrupt 7
4	PORTE	Port E pin 7 selected for external interrupt 7
5	PORTF	Port F pin 7 selected for external interrupt 7

27	Reserved	To ensure compatibility with future devices, always write bits to 0.		
26:24	EXTIPSEL6	0x0	RW	**External Interrupt 6 Port Select**
	Select input port for external interrupt 6.			

Value	Mode	Description
0	PORTA	Port A pin 6 selected for external interrupt 6
1	PORTB	Port B pin 6 selected for external interrupt 6
2	PORTC	Port C pin 6 selected for external interrupt 6
3	PORTD	Port D pin 6 selected for external interrupt 6
4	PORTE	Port E pin 6 selected for external interrupt 6
5	PORTF	Port F pin 6 selected for external interrupt 6

23	Reserved	To ensure compatibility with future devices, always write bits to 0.		
22:20	EXTIPSEL5	0x0	RW	**External Interrupt 5 Port Select**
	Select input port for external interrupt 5.			

Bit	Name	Reset	Access	Description

Value	Mode	Description
0	PORTA	Port A pin 5 selected for external interrupt 5
1	PORTB	Port B pin 5 selected for external interrupt 5
2	PORTC	Port C pin 5 selected for external interrupt 5
3	PORTD	Port D pin 5 selected for external interrupt 5
4	PORTE	Port E pin 5 selected for external interrupt 5
5	PORTF	Port F pin 5 selected for external interrupt 5

Bit	Name	Reset	Access	Description
19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18:16	EXTIPSEL4	0x0	RW	**External Interrupt 4 Port Select**

Select input port for external interrupt 4.

Value	Mode	Description
0	PORTA	Port A pin 4 selected for external interrupt 4
1	PORTB	Port B pin 4 selected for external interrupt 4
2	PORTC	Port C pin 4 selected for external interrupt 4
3	PORTD	Port D pin 4 selected for external interrupt 4
4	PORTE	Port E pin 4 selected for external interrupt 4
5	PORTF	Port F pin 4 selected for external interrupt 4

Bit	Name	Reset	Access	Description
15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14:12	EXTIPSEL3	0x0	RW	**External Interrupt 3 Port Select**

Select input port for external interrupt 3.

Value	Mode	Description
0	PORTA	Port A pin 3 selected for external interrupt 3
1	PORTB	Port B pin 3 selected for external interrupt 3
2	PORTC	Port C pin 3 selected for external interrupt 3
3	PORTD	Port D pin 3 selected for external interrupt 3
4	PORTE	Port E pin 3 selected for external interrupt 3
5	PORTF	Port F pin 3 selected for external interrupt 3

Bit	Name	Reset	Access	Description
11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	EXTIPSEL2	0x0	RW	**External Interrupt 2 Port Select**

Select input port for external interrupt 2.

Value	Mode	Description
0	PORTA	Port A pin 2 selected for external interrupt 2
1	PORTB	Port B pin 2 selected for external interrupt 2
2	PORTC	Port C pin 2 selected for external interrupt 2
3	PORTD	Port D pin 2 selected for external interrupt 2
4	PORTE	Port E pin 2 selected for external interrupt 2
5	PORTF	Port F pin 2 selected for external interrupt 2

Bit	Name	Reset	Access	Description
7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:4	EXTIPSEL1	0x0	RW	**External Interrupt 1 Port Select**

Select input port for external interrupt 1.

Value	Mode	Description
0	PORTA	Port A pin 1 selected for external interrupt 1
1	PORTB	Port B pin 1 selected for external interrupt 1
2	PORTC	Port C pin 1 selected for external interrupt 1
3	PORTD	Port D pin 1 selected for external interrupt 1
4	PORTE	Port E pin 1 selected for external interrupt 1
5	PORTF	Port F pin 1 selected for external interrupt 1

Bit	Name	Reset	Access	Description
3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	EXTIPSEL0	0x0	RW	**External Interrupt 0 Port Select**

Select input port for external interrupt 0.

Bit	Name	Reset	Access	Description
	Value	**Mode**		**Description**
	0	PORTA		Port A pin 0 selected for external interrupt 0
	1	PORTB		Port B pin 0 selected for external interrupt 0
	2	PORTC		Port C pin 0 selected for external interrupt 0
	3	PORTD		Port D pin 0 selected for external interrupt 0
	4	PORTE		Port E pin 0 selected for external interrupt 0
	5	PORTF		Port F pin 0 selected for external interrupt 0

AF.5.11 GPIO_EXTIPSELH - External Interrupt Port Select High Register

Offset	Bit Position																														
0x104	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0				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	Reserved	To ensure compatibility with future devices, always write bits to 0.		
30:28	EXTIPSEL15	0x0	RW	**External Interrupt 15 Port Select**
	Select input port for external interrupt 15.			
	Value	**Mode**		**Description**
	0	PORTA		Port A pin 15 selected for external interrupt 15
	1	PORTB		Port B pin 15 selected for external interrupt 15
	2	PORTC		Port C pin 15 selected for external interrupt 15
	3	PORTD		Port D pin 15 selected for external interrupt 15
	4	PORTE		Port E pin 15 selected for external interrupt 15
	5	PORTF		Port F pin 15 selected for external interrupt 15
27	Reserved	To ensure compatibility with future devices, always write bits to 0.		
26:24	EXTIPSEL14	0x0	RW	**External Interrupt 14 Port Select**
	Select input port for external interrupt 14.			
	Value	**Mode**		**Description**
	0	PORTA		Port A pin 14 selected for external interrupt 14
	1	PORTB		Port B pin 14 selected for external interrupt 14
	2	PORTC		Port C pin 14 selected for external interrupt 14
	3	PORTD		Port D pin 14 selected for external interrupt 14
	4	PORTE		Port E pin 14 selected for external interrupt 14
	5	PORTF		Port F pin 14 selected for external interrupt 14
23	Reserved	To ensure compatibility with future devices, always write bits to 0.		
22:20	EXTIPSEL13	0x0	RW	**External Interrupt 13 Port Select**
	Select input port for external interrupt 13.			
	Value	**Mode**		**Description**
	0	PORTA		Port A pin 13 selected for external interrupt 13
	1	PORTB		Port B pin 13 selected for external interrupt 13
	2	PORTC		Port C pin 13 selected for external interrupt 13
	3	PORTD		Port D pin 13 selected for external interrupt 13
	4	PORTE		Port E pin 13 selected for external interrupt 13
	5	PORTF		Port F pin 13 selected for external interrupt 13
19	Reserved	To ensure compatibility with future devices, always write bits to 0.		
18:16	EXTIPSEL12	0x0	RW	**External Interrupt 12 Port Select**

Bit	Name	Reset	Access	Description

Select input port for external interrupt 12.

Value	Mode	Description
0	PORTA	Port A pin 12 selected for external interrupt 12
1	PORTB	Port B pin 12 selected for external interrupt 12
2	PORTC	Port C pin 12 selected for external interrupt 12
3	PORTD	Port D pin 12 selected for external interrupt 12
4	PORTE	Port E pin 12 selected for external interrupt 12
5	PORTF	Port F pin 12 selected for external interrupt 12

15	Reserved	To ensure compatibility with future devices, always write bits to 0.		
14:12	EXTIPSEL11	0x0	RW	**External Interrupt 11 Port Select**

Select input port for external interrupt 11.

Value	Mode	Description
0	PORTA	Port A pin 11 selected for external interrupt 11
1	PORTB	Port B pin 11 selected for external interrupt 11
2	PORTC	Port C pin 11 selected for external interrupt 11
3	PORTD	Port D pin 11 selected for external interrupt 11
4	PORTE	Port E pin 11 selected for external interrupt 11
5	PORTF	Port F pin 11 selected for external interrupt 11

11	Reserved	To ensure compatibility with future devices, always write bits to 0.		
10:8	EXTIPSEL10	0x0	RW	**External Interrupt 10 Port Select**

Select input port for external interrupt 10.

Value	Mode	Description
0	PORTA	Port A pin 10 selected for external interrupt 10
1	PORTB	Port B pin 10 selected for external interrupt 10
2	PORTC	Port C pin 10 selected for external interrupt 10
3	PORTD	Port D pin 10 selected for external interrupt 10
4	PORTE	Port E pin 10 selected for external interrupt 10
5	PORTF	Port F pin 10 selected for external interrupt 10

7	Reserved	To ensure compatibility with future devices, always write bits to 0.		
6:4	EXTIPSEL9	0x0	RW	**External Interrupt 9 Port Select**

Select input port for external interrupt 9.

Value	Mode	Description
0	PORTA	Port A pin 9 selected for external interrupt 9
1	PORTB	Port B pin 9 selected for external interrupt 9
2	PORTC	Port C pin 9 selected for external interrupt 9
3	PORTD	Port D pin 9 selected for external interrupt 9
4	PORTE	Port E pin 9 selected for external interrupt 9
5	PORTF	Port F pin 9 selected for external interrupt 9

3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2:0	EXTIPSEL8	0x0	RW	**External Interrupt 8 Port Select**

Select input port for external interrupt 8.

Value	Mode	Description
0	PORTA	Port A pin 8 selected for external interrupt 8
1	PORTB	Port B pin 8 selected for external interrupt 8
2	PORTC	Port C pin 8 selected for external interrupt 8
3	PORTD	Port D pin 8 selected for external interrupt 8
4	PORTE	Port E pin 8 selected for external interrupt 8
5	PORTF	Port F pin 8 selected for external interrupt 8

AF.5.12 GPIO_EXTIRISE - External Interrupt Rising Edge Trigger 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

Offset	Bit Position	
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.		
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

AF.5.13 GPIO_EXTIFALL - External Interrupt Falling Edge Trigger 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																	0x0000															
Access																	RW															
Name																	EXTIFALL															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
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

AF.5.14 GPIO_IEN - Interrupt Enable 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																	0x0000															
Access																	RW															
Name																	EXT															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	EXT	0x0000	RW	**External Interrupt n Enable**

Set bit n to enable external interrupt from pin n.

Value	Description
EXT[n] = 0	Pin n external interrupt disabled
EXT[n] = 1	Pin n external interrupt enabled

AF.5.15 GPIO_IF - Interrupt Flag 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	0x0000																															
Access	R																															
Name	EXT																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	EXT	0x0000	R	**External Interrupt Flag n**

Pin n external interrupt flag.

Value	Description
EXT[n] = 0	Pin n external interrupt flag cleared
EXT[n] = 1	Pin n external interrupt flag set

AF.5.16 GPIO_IFS - Interrupt Flag Set 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	0x0000																															
Access	W1																															
Name	EXT																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	EXT	0x0000	W1	**External Interrupt Flag n Set**

Write bit n to 1 to set interrupt flag n.

Value	Description
EXT[n] = 0	Pin n external interrupt flag unchanged
EXT[n] = 1	Pin n external interrupt flag set

AF.5.17 GPIO_IFC - Interrupt Flag Clear 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	0x0000																															
Access	W1																															
Name	EXT																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	EXT	0x0000	W1	**External Interrupt Flag Clear**

Write bit n to 1 to clear external interrupt flag n.

Value	Description
EXT[n] = 0	Pin n external interrupt flag unchanged
EXT[n] = 1	Pin n external interrupt flag cleared

AF.5.18 GPIO_ROUTE - I/O Routing Register

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									0	0	0	0	0			0x0							0	1	1
Access							RW									RW	RW	RW	RW	RW			RW							RW	RW	RW
Name							ETMLOCATION									TD3PEN	TD2PEN	TD1PEN	TD0PEN	TCLKPEN			SWLOCATION							SWOPEN	SWDIOPEN	SWCLKPEN

Bit	Name	Reset	Access	Description
31:26	Reserved	To ensure compatibility with future devices, always write bits to 0.		
25:24	ETMLOCATION	0x0	RW	**I/O Location**
	Decides the location of the TCLK and TD 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
23:17	Reserved	To ensure compatibility with future devices, always write bits to 0.		
16	TD3PEN	0	RW	**ETM Trace Data Pin Enable**
	Enable ETM Trace Data Output 3 connection to pin.			
15	TD2PEN	0	RW	**ETM Trace Data Pin Enable**
	Enable ETM Trace Data Output 2 connection to pin.			
14	TD1PEN	0	RW	**ETM Trace Data Pin Enable**
	Enable ETM Trace Data Output 1 connection to pin.			
13	TD0PEN	0	RW	**ETM Trace Data Pin Enable**
	Enable ETM Trace Data Output 0 connection to pin.			
12	TCLKPEN	0	RW	**ETM Trace Clock Pin Enable**
	Enable ETM Trace Clock Output connection to pin.			
11:10	Reserved	To ensure compatibility with future devices, always write bits to 0.		
9:8	SWLOCATION	0x0	RW	**I/O Location**
	Decides the location of the SW 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
7:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	SWOPEN	0	RW	**Serial Wire Viewer Output Pin Enable**
	Enable Serial Wire Viewer Output connection to pin.			
1	SWDIOPEN	1	RW	**Serial Wire Data Pin Enable**
	Enable Serial Wire Data 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	SWCLKPEN	1	RW	**Serial Wire Clock Pin Enable**
	Enable Serial Wire 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.			

AF.5.19 GPIO_INSENSE - Input Sense Register

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

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	PRS	1	RW	**PRS Sense Enable**
	Set this bit to enable input sensing for PRS.			
0	INT	1	RW	**Interrupt Sense Enable**
	Set this bit to enable input sensing for interrupts.			

AF.5.20 GPIO_LOCK - Configuration Lock Register

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																	0x0000															
Access																	RW															
Name																	LOCKKEY															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0.		
15:0	LOCKKEY	0x0000	RW	**Configuration Lock Key**
	Write any other value than the unlock code to lock MODEL, MODEH, CTRL, PINLOCKN, EPISELL, EIPSELH, INSENSE and SWDPROUTE 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

AF.5.21 GPIO_CTRL - GPIO Control Register

Offset	Bit Position																													
0x12C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																																0
Access																																RW
Name																																EM4RET

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	EM4RET	0	RW	**Enable EM4 retention**
	Set to enable EM4 retention of output enable, output value and pull enable.			

AF.5.22 GPIO_CMD - GPIO Command Register

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																																0
Access																																W1
Name																																EM4WUCLR

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	EM4WUCLR	0	W1	**EM4 Wake-up clear**
	Write 1 to clear all wake-up requests.			

AF.5.23 GPIO_EM4WUEN - EM4 Wake-up Enable Register

Offset	Bit Position																													
0x134	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x00				
Access																											RW				
Name																											EM4WUEN				

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:0	EM4WUEN	0x00	RW	**EM4 Wake-up enable**
	Write 1 to enable wake-up request, write 0 to disable wake-up request.			

Value	Mode	Description
0x01	A0	Enable em4 wakeup on pin A0
0x02	A6	Enable em4 wakeup on pin A6
0x04	C9	Enable em4 wakeup on pin C9
0x08	F1	Enable em4 wakeup on pin F1
0x10	F2	Enable em4 wakeup on pin F2
0x20	E13	Enable em4 wakeup on pin E13

AF.5.24 GPIO_EM4WUPOL - EM4 Wake-up Polarity Register

Offset	Bit Position																													
0x138	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																											0x00				
Access																											RW				
Name																											EM4WUPOL				

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:0	EM4WUPOL	0x00	RW	**EM4 Wake-up Polarity**

Write bit n to 1 for high wake-up request. Write bit n to 0 for low wake-up request

Value	Mode	Description
0x01	A0	Determines polarity on pin A0
0x02	A6	Determines polarity on pin A6
0x04	C9	Determines polarity on pin C9
0x08	F1	Determines polarity on pin F1
0x10	F2	Determines polarity on pin F2
0x20	E13	Determines polarity on pin E13

AF.5.25 GPIO_EM4WUCAUSE - EM4 Wake-up Cause Register

Offset	Bit Position																													
0x13C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0x00																													
Access	R																													
Name	EM4WUCAUSE																													

Bit	Name	Reset	Access	Description
31:6	Reserved	To ensure compatibility with future devices, always write bits to 0.		
5:0	EM4WUCAUSE	0x00	R	**EM4 wake-up cause**

Bit n indicates which pin the wake-up request occurred.

Value	Mode	Description
0x01	A0	This bit indicates an em4 wake-up request occurred on pin A0
0x02	A6	This bit indicates an em4 wake-up request occurred on pin A6
0x04	C9	This bit indicates an em4 wake-up request occurred on pin C9
0x08	F1	This bit indicates an em4 wake-up request occurred on pin F1
0x10	F2	This bit indicates an em4 wake-up request occurred on pin F2
0x20	E13	This bit indicates an em4 wake-up request occurred on pin E13

AG ARM Universal Serial Bus Controller

AG.1 Introduction

The USB is a full-speed/low-speed USB 2.0 compliant OTG host/device controller. The architecture is very flexible and allows the USB to be used in On-the-go (OTG) Dual-Role Device, Device and Host-only configurations. The USB supports HNP and SRP protocols and both OTG Revisions 1.3 and 2.0 are supported. The on-chip voltage regulator and PHY reduces the number of external components to a minimum. A switchable external 5V supply or step-up regulator is needed for OTG Dual Role Device and Host configurations.

AG.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
- 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)
- SRP detection in EM2 (during host session off)
- Soft connect/disconnect
- Full OTG support
  - Compliant with On-The-Go and Embedded Host Supplement to the USB Revision 2.0 Specification, Revision 2.0
  - Compliant with USB On-The-Go Supplement, Revision 1.3
  - Supports Host Negotiation Protocol (HNP) and Session Request Protocol (SRP)
- On-chip PHY
  - Internal pull-up and pull-down resistors
  - Voltage comparators for monitoring VBUS voltage
  - A/B Device identification using ID line
  - Charge/discharge of VBUS for VBUS-pulsing
- Internal 3.3V Regulator
  - Output voltage: 3.3V
  - Output current: 50 mA
  - Input voltage range: 4.0 - 5.5V
  - Enabled automatically when input voltage applied

- Low quiescent current: 100 uA
- Dedicated input pin allows regulator to be used in OTG and host configurations
- Output pin can be used to power the EFM32 itself as well as external components
- Regulator voltage output sense feature for detecting USB plug/unplug events (also available in EM2/3)

AG.3 USB System Description

An block diagram of the USB is shown in Figure 320.

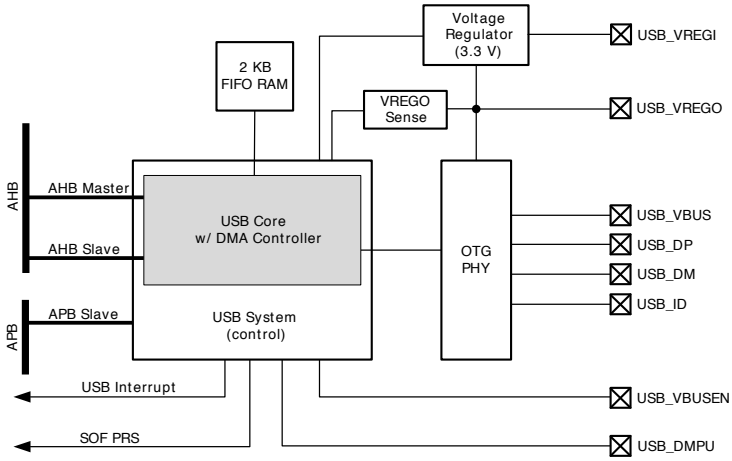


Figure 320: USB Block Diagram

The USB consists of a digital logic part, a 2 KB endpoint RAM, OTG 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 itself as well as external components.

The digital logic of the USB is split into two parts: system and core.

The system part is accessed using USB registers from offset 0x000 to 0x018 and controls the voltage regulator and enabling/disabling of the PHY and USB pins. This part is clocked by $HFCORECLK_{USB}$ and is accessed using an APB slave interface. The system part can thus be accessed independently of the core part, without $HFCORECLK_{USBC}$ running.

The core part is clocked by $HFCORECLK_{USBC}$ 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 0x3C000. 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 clock to the system part must not be disabled when the core part is active.

There are 8 pins associated with the USB. USB_VBUS should be connected to the VBUS (5V) pin on the USB receptacle. It is connected to the voltage comparators and current sink/source in the PHY. USB_DP and USB_DM are the USB D+ and D- pins. These are the USB data signaling pins. USB_ID is the OTG ID pin used to detect the device type (A or B). This pin can be left unconnected when not used. USB_VREGI is the input to the voltage regulator and USB_VREGO is the regulated output. USB_VBUSEN is used to turn on and off VBUS power when operating as host-only or OTG A-Device. USB_DMPU is used to enable/disable an external D- pull-up resistor. This is needed for low-speed device only. USB_VBUSEN and USB_DMPU will be high-impedance until the pins are enabled from software. Thus, if a defined level is required during start-up an external pull-up/pull-down can be used.

AG.3.1 USB Initialization

The USB requires the device to run from a 48 MHz crystal (2500 ppm or better). The core part of the USB will always run from $HFCORECLK_{USBC}$ which is HFCLK undivided (48 MHz). The current consumption for the rest of the device can be reduced by dividing down HFCORECLK using the CMU_HFCORECLKDIV register. Bandwidth requirements for the specific USB application must be taken into account when dividing down HFCORECLK.

Follow these steps to enable the USB:

1. Enable the clock to the system part by setting USB in CMU_HFCORECLKEN0.
2. If the internal USB regulator is bypassed (by applying 3.3V on USB_VREGI and USB_VREGO externally), disable the regulator by setting VREGDIS in USB_CTRL.
3. If the PHY is powered from VBUS using the internal regulator, the VREGO sense circuit should be enabled by setting VREGOSEN in USB_CTRL.
4. Enable the USB PHY pins by setting PHYPEN in USB_ROUTE.
5. If host or OTG dual-role device, 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. If low-speed device, set DMPUAP in USB_CTRL to the desired value and then enable the USB_DMPU pin in USB_ROUTE. Set the MODE for the pin to PUSHPULL.
7. Make sure HFXO is ready and selected. The core part requires the undivided HFCLK to be 48 MHz when USB is active (during suspend/session-off a 32 kHz clock is used)..
8. Enable the clock to the core part by setting USBC in CMU_HFCORECLKEN0.
9. Wait for the core to come out of reset. This is easiest done by polling a core register with non-zero reset value until it reads a non-zero value. This takes approximately 20 48-MHz cycles.
10. Start initializing the USB core as described in USB Core Description.

AG.3.2 Configurations

The USB can be used as Device, OTG Dual Role 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 4.7 uF external decoupling capacitor on the input and a 1 uF external decoupling capacitor on the output. Decoupling not related to USB is not shown in the figures.

Bus-powered Device

A bus-powered device configuration is shown in Figure 321. In this configuration the voltage regulator powers the PHY and the EFM32 at 3.3 V. The voltage regulator output (USB_VREGO) can also be used to power other components of the system.

In this configuration, the VREGO sense circuit should be left disabled.

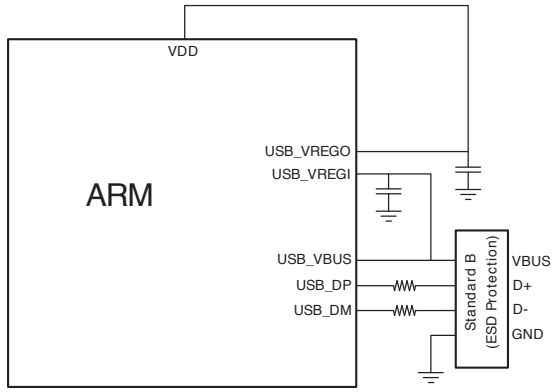


Figure 321: Bus-powered Device

Self-powered Device

A self-powered device configuration is shown in Figure 322. When the USB is configured as a self-powered device, the voltage regulator is typically used to power the PHY only, although it may also be used to power other 3.3 V components. When the USB is connected to a host, the voltage regulator is activated. Software can detect this event by enabling the VREGO Sense High (VREGOSH) interrupt. The PHY pins can then be enabled and USB traffic can start. The VREGO Sense Low (VREGOSL) interrupt can be used to detect when VBUS voltage disappears (for example if the USB cable is unplugged).

In this configuration, the VREGO sense circuit must be enabled.

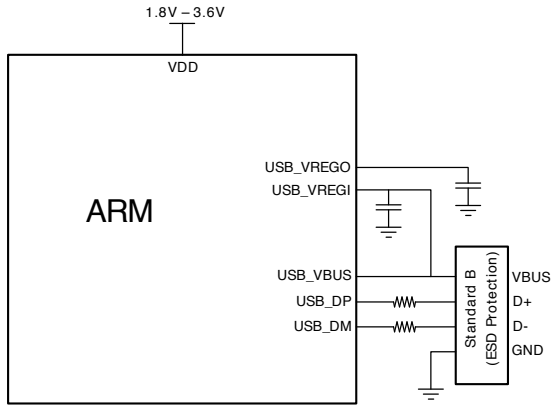


Figure 322: Self-powered Device

Self-powered Device (with bus-power switch)

A self-powered device (with bus-power switch) may switch power supply to VBUS when connected to a host. This is typically useful for extending the life of battery-powered devices and enables the use of coin-cell driven systems with low maximum peak current. The external components required typically include 2 transistors, 2 diodes and a few resistors. See application note for details. This allows seamless power supply switching between a battery and the voltage regulator output.

The VREGO Sense High interrupt is used to detect when VBUS becomes present. Software can then enable the external transistor connected to USB_VREGO, effectively switching the power source. A regular GPIO pin is used to control this transistor. If necessary, the application may have to reduce the current consumption before switching to the USB power source. If VBUS voltage is removed, the circuit switches automatically back to the battery power supply. If necessary software must react quickly to this event and reduce the current consumption (for example by reducing the clock frequency) to avoid excessive voltage drop. This configuration is shown in Figure 323.

In this configuration, the VREGO sense circuit must be enabled.

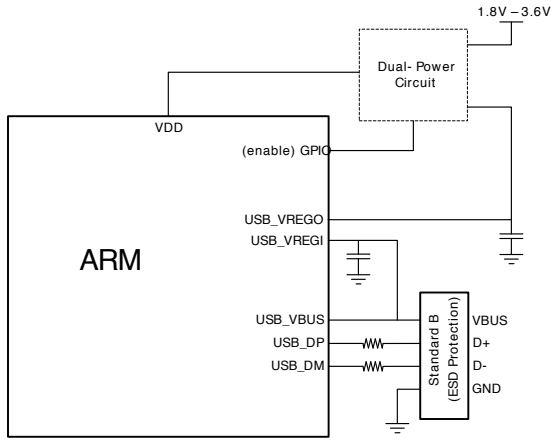


Figure 323: Self-powered Device (with bus-power switch)

OTG Dual Role Device (5V)

An OTG Dual Role Device (5V) configuration is shown in Figure 324. When 5V is available, the internal regulator can be used to power the EFM32. An external power switch is needed to control VBUS power. For over-current detection a regular GPIO input pin with interrupt is used. The application should turn off or limit VBUS power when over-current is detected. In OTG mode, the maximum VBUS decoupling capacitance is 6.5 uF.

In this configuration, the VREGO sense circuit should be left disabled.

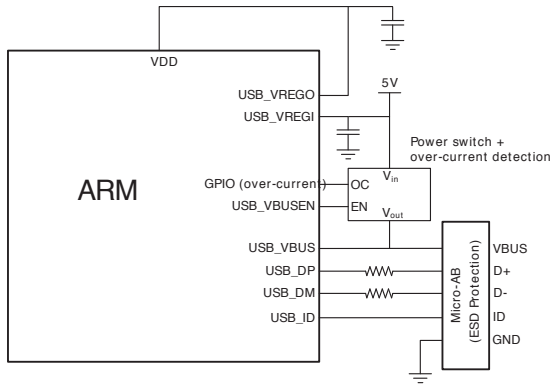


Figure 324: OTG Dual Role Device (5V)

OTG Dual Role Device (5V step-up regulator)

An OTG Dual Role Device (5V step-up regulator) configuration is shown in Figure 325. When 5V is not available, an external 5V step-up regulator is needed. In this configuration, the voltage for the EFM32 must be in the range 3.0V - 3.6V. In this mode the voltage regulator is bypassed by connecting both the input and out-

put to the external supply. This effectively causes the PHY to be powered directly from the external 3.0 - 3.6 V supply. The voltage regulator should be disabled when operating in this mode. For over-current detection a regular GPIO input pin with interrupt is used. The application should turn off or limit VBUS power when over-current is detected. In OTG mode, the maximum VBUS decoupling capacitance is 6.5 uF.

In this configuration, the VREGO sense circuit should be left disabled.

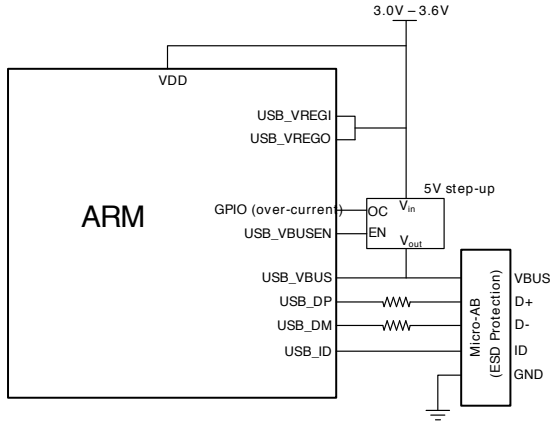


Figure 325: OTG Dual Role Device (5V step-up regulator)

Host

A host configuration is shown in Figure 326. In this example a 5V step-up regulator is used. If 5V is available, a power switch can be used instead, as shown in Figure 324. The host configuration is equal to OTG Dual Role Device, except for the USB_ID pin which is not used and the USB connector which is a USB Standard-A Connector. In host mode, the minimum VBUS decoupling capacitance is 96 uF.

In this configuration, the VREGO sense circuit should be left disabled.

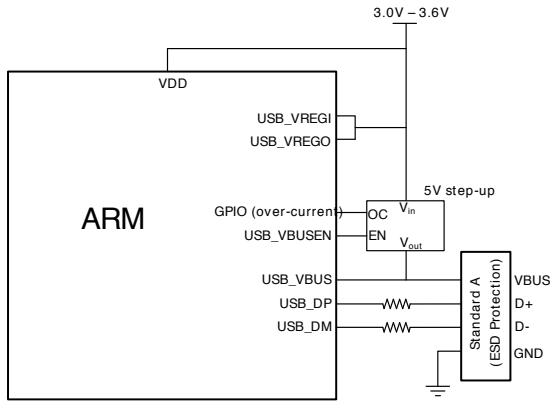


Figure 326: Host

AG.3.3 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 USB core clock is enabled.

The PHY is powered by the internal voltage regulator output (USB_VREGO). To power the PHY directly from an external source (for example an external 3.3 V LDO), connect both USB_VREGO and USB_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. Also, the PHY must always be disabled manually when there is no voltage applied to VREGO.

AG.3.4 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 USB_VREGI and USB_VREGO. Note that the USB standard requires the total capacitance on VBUS to be 1 uF minimum and 10 uF maximum for regular devices. OTG devices can have maximum 6.5 uF capacitance on VBUS.

The voltage regulator is enabled by default and can thus be used to power the EFM32 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 (typically due to the USB cable being unplugged). If regulator voltage monitoring is not required (i.e. it is known that the VREGO voltage is always present), the sense circuit should be left disabled.

During suspend, the bias current for the regulator can be reduced if the current requirements in EM2/3 are low. The bias current in EM2/3 is controlled by BIASPROGEM23 in USB_CTRL. When EM2/3 is entered, the bias current for the regulator switches to what is specified in BIASPROGEM23 in USB_CTRL. When entering EM0 again (due to USB resume/reset signaling or any other wake-up interrupt) the regulator switches back to using the value specified in BIASPROGEM01 in USB_CTRL.

AG.3.5 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.

AG.3.6 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, $HFCORECLK_{USBC}$ must be switched from 48 MHz to 32 kHz (LFXO or LFRCO). This is done using the CMU_CMD and CMU_STATUS registers. While $HFCORECLK_{USBC}$ is 32 kHz, the USB core registers (starting from offset 0x3C000) 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, $HFCORECLK_{USBC}$ must be switched back to 48 MHz before accessing the core registers. The device always starts up from HFRCO so software must restart HFXO and switch from HFRCO to HFXO. The USB system clock, $HFCORECLK_{USB}$, 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.

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. VBUS becomes 5V).

AG.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:

- HNP- and SRP-Capable OTG (Device and Host)
- 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
- ADP is not supported

AG.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 Section AG.6.

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 HS PHY or Internal FS 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 Section AG.4.1 or Device Initialization sequence.

Note

The core is designed to be interrupt-driven. Polling interrupt mechanism is not recommended: this may result in undefined resolutions.

Note

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.

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 Device Initialization.

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.CONIDSTSCHN 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.

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.

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
   - Non-Zero-Length Status OUT Handshake
   - Periodic Frame Interval

2. Program the USB_GINTMSK register to unmask the following interrupts.
   - USB Reset
   - Enumeration Done
   - Early Suspend
   - USB Suspend
3. 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 Initialization on USB Reset
4. 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 Initialization on Enumeration Completion

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

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.

When 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.

When 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 in OTG Revision 1.3 mode. 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. The core perform data-line pulsing followed by VBUS pulsing.
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.

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.

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 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 Device Initialization.

AG.4.2 Modes of operation

- Overview: DMA/Slave modes
- DMA Mode
- Slave Mode

Overview: DMA/Slave modes

The application can operate the core in either of two modes:

- In DMA Mode - The core fetches the data to be transmitted or updates the received data on the AHB.
- In Slave Mode — The application initiates the data transfers for data fetch and store.

DMA Mode

In DMA Mode, the OTG 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.

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.

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).

Slave Mode

In Slave mode, the application can operate the core either in transaction-level (packet-level) operation or in pipelined transaction-level operation.

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 327 and Figure 328.

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.

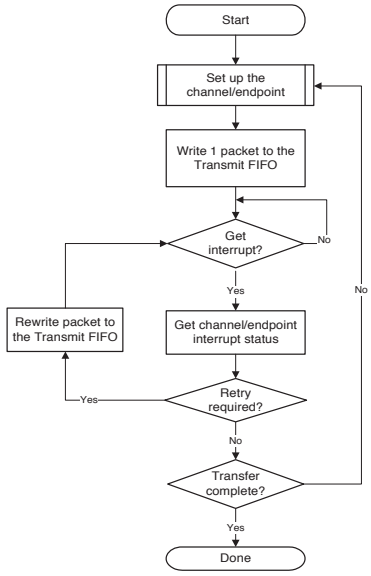


Figure 327: Transmit Transaction-Level Operation in Slave Mode

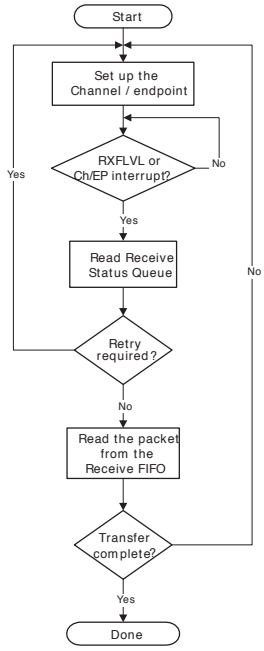


Figure 328: Receive Transaction-Level Operation in Slave Mode

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.

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.

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 last 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.

AG.4.3 Host Programming Model

Before you program the Host, read Overview: Programming the Core and Modes of operation.

This section discusses the following topics:

- Channel Initialization
- Halting a Channel
- Zero-Length Packets
- Handling Babble Conditions
- Handling Disconnects
- Host Programming Operations
  - Writing the Transmit FIFO in Slave Mode
  - Reading the Receive FIFO in Slave Mode

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.

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.

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.

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.

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.

Host Programming Operations [?] provides links to the programming sequence for the different types of USB transactions.

Writing the Transmit FIFO in Slave Mode Figure 329 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.

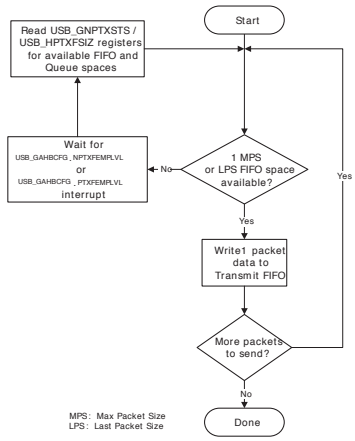


Figure 329: Transmit FIFO Write Task in Slave Mode

Reading the Receive FIFO in Slave Mode Figure 330 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).

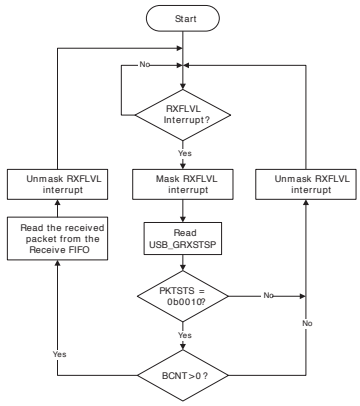


Figure 330: Receive FIFO Read Task in Slave Mode

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 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 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.

Bulk and Control OUT/SETUP Transactions in Slave Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization. See Figure 329 and Figure 330 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 331. 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.

Normal Bulk and Control OUT/SETUP Operations The sequence of operations in Figure 331 (channel 1) is as follows:

1. Initialize channel 1 as explained in 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.

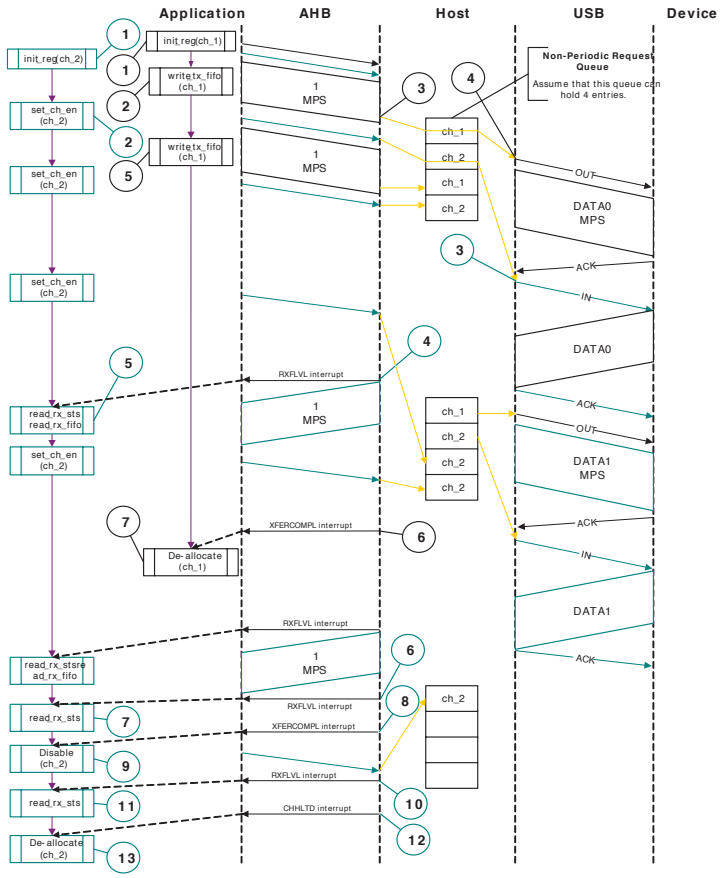


Figure 331: Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in Slave Mode

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.

Bulk and Control IN Transactions in Slave Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization. See Figure 329 and Figure 330 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 331. 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.

Normal Bulk and Control IN Operations The sequence of operations in Figure 331 is as follows:

1. Initialize channel 2 as explained in 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 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.

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
}
```

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 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 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.

Bulk and Control OUT/SETUP Transactions in DMA Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization.

This section discusses the following topics:

- Overview
- Normal Bulk and Control OUT/SETUP Operations
- NAK Handling with DMA
- Handling Interrupts

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.

Normal Bulk and Control OUT/SETUP Operations The sequence of operations in Figure 331 is as follows:

1. Initialize and enable channel 1 as explained in 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 Handling Interrupts.

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.

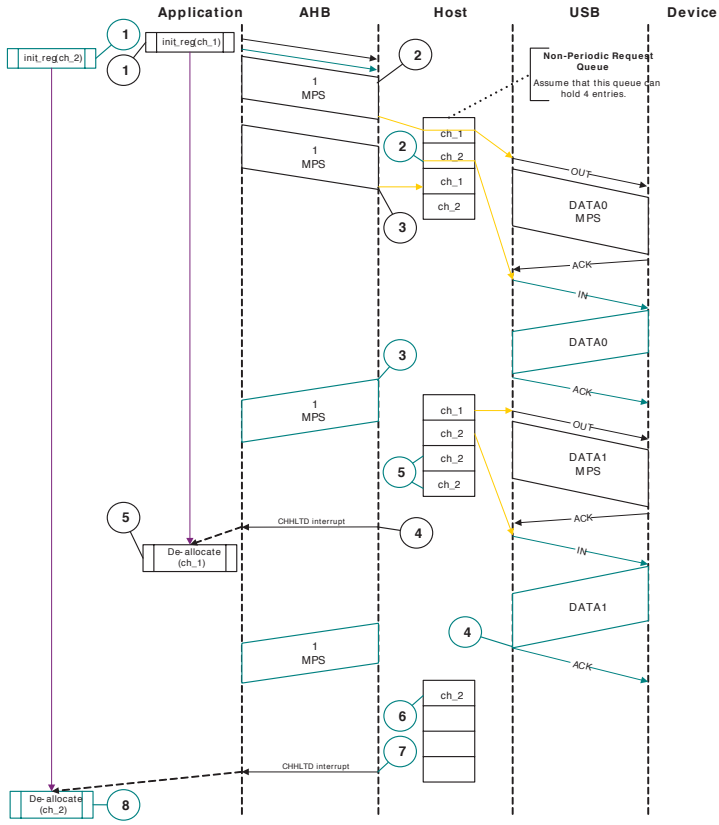


Figure 332: Normal Bulk/Control OUT/SETUP and Bulk/Control IN Transactions in DMA Mode

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.

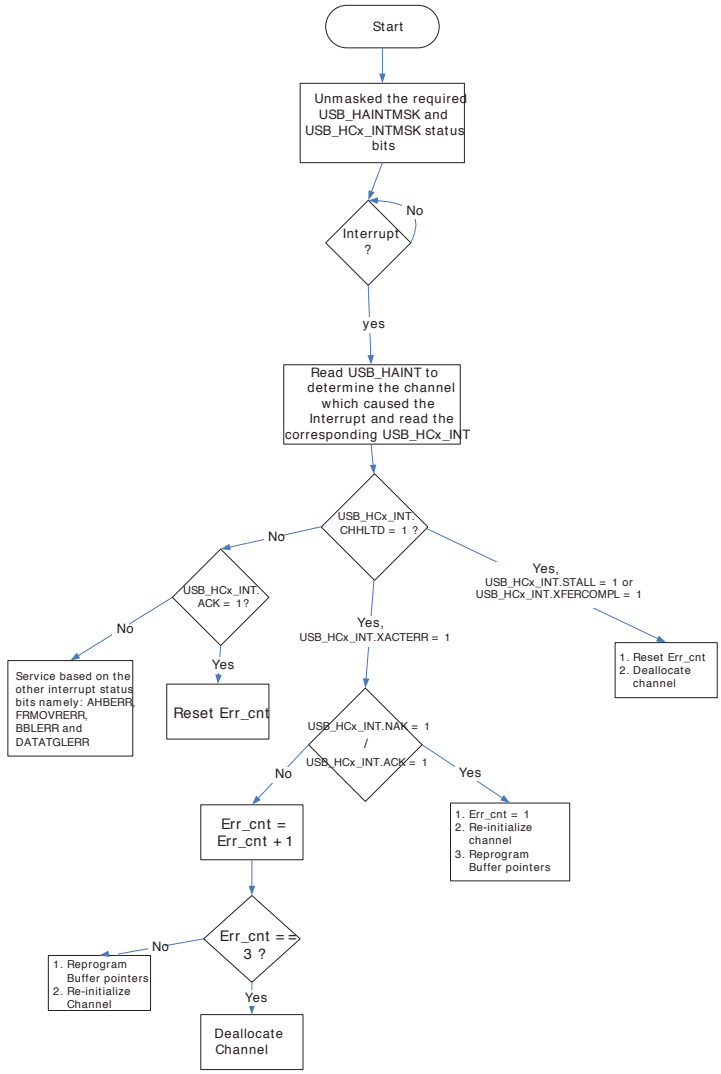


Figure 333: Interrupt Service Routine for Bulk/Control OUT Transaction in DMA Mode

In Figure 333 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.

Bulk and Control IN Transactions in DMA Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization.

A typical bulk or control IN operation in DMA mode is shown in Figure 332. 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.

Normal Bulk and Control IN Operations The sequence of operations in Figure 332 is as follows:

1. Initialize and enable channel 2 as explained in 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.

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
}
```

Interrupt OUT Transactions in Slave Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization. See Figure 329 and Figure 330 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 334. 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.

Normal Interrupt OUT Operation The sequence of operations in Figure 334 is as follows:

1. Initialize and enable channel 1 as explained in 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.

Handling Interrupts The channel-specific interrupt service routine for Interrupt OUT transactions in Slave mode is shown in the following flow:

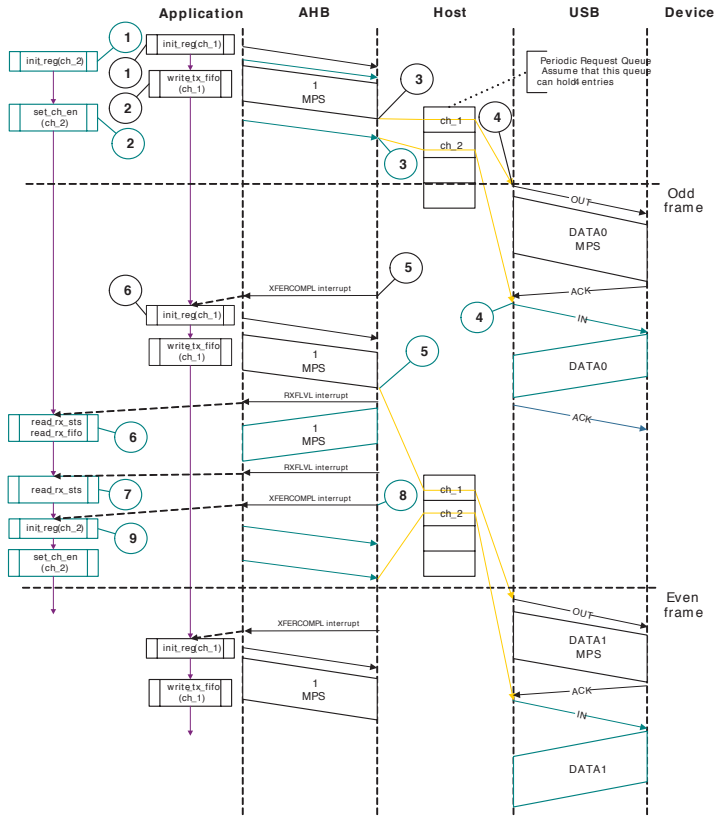


Figure 334: Normal Interrupt OUT/IN Transactions in Slave Mode

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.

Interrupt IN Transactions in Slave Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in 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 334. 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.

Normal Interrupt IN Operation The sequence of operations in Figure 334 (channel 2) is as follows:

1. Initialize channel 2 as explained in 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 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.

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).

Interrupt OUT Transactions in DMA Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization.

A typical interrupt OUT operation in DMA mode is shown in Figure 335. 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.

Normal Interrupt OUT Operation

1. Initialize and enable channel 1 as explained in 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.

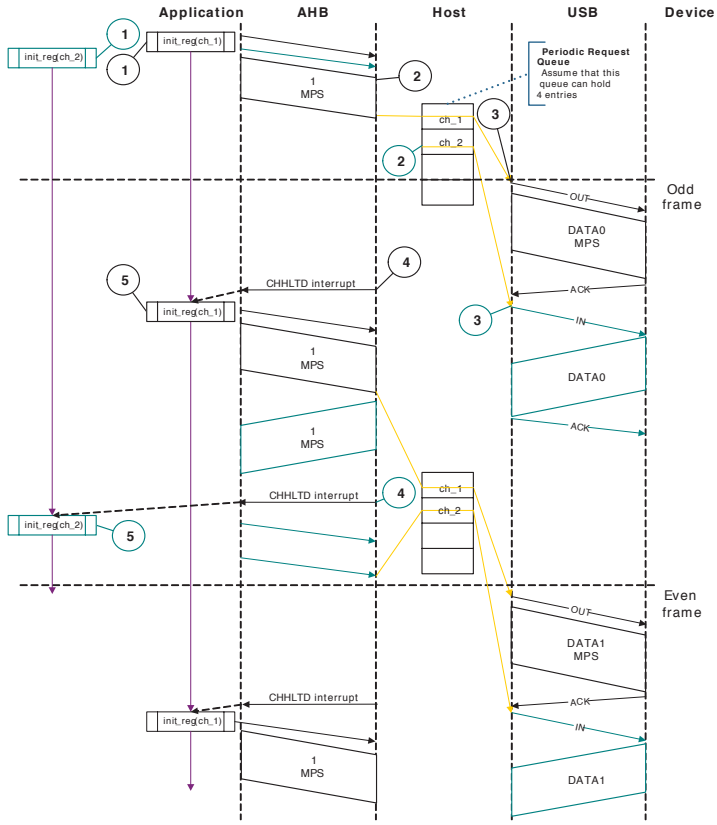


Figure 335: Normal Interrupt OUT/IN Transactions in DMA Mode

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).

Interrupt IN Transactions in DMA Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization.

A typical interrupt IN operation in DMA mode is shown in Figure 335. 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.

Normal Interrupt IN Operation The sequence of operations in Figure 335 (channel 2) is as follows:

1. Initialize and enable channel 2 as explained in 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.

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.

Isochronous OUT Transactions in Slave Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization. See TFigure 329 and Figure 330 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 336. 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.

Normal Isochronous OUT Operation The sequence of operations in Figure 336 (channel 1) is as follows:

1. Initialize and enable channel 1 as explained in 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.

Handling Interrupts The channel-specific interrupt service routine for isochronous OUT transactions in Slave mode is shown in the following flow:

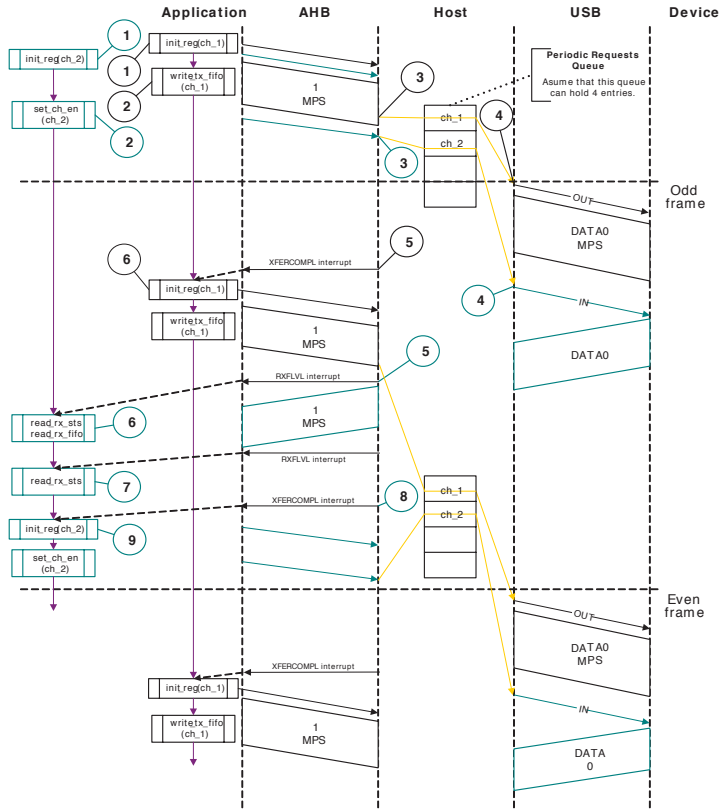


Figure 336: Normal Isochronous OUT/IN Transactions in Slave Mode

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
```

```
}
```

Isochronous IN Transactions in Slave Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization. See Figure 329 and Figure 330 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 336. 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.

Normal Isochronous IN Operation The sequence of operations in Figure 336 (channel 2) is as follows:

1. Initialize channel 2 as explained in 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 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.

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
    }
}
```

Isochronous OUT Transactions in DMA Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the

Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization.

A typical isochronous OUT operation in DMA mode is shown in Figure 337. 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.

Normal Isochronous OUT Operation

1. Initialize and enable channel 1 as explained in 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.

Handling Interrupts The channel-specific interrupt service routine for Isochronous OUT transactions in DMA mode is shown in the following flow:

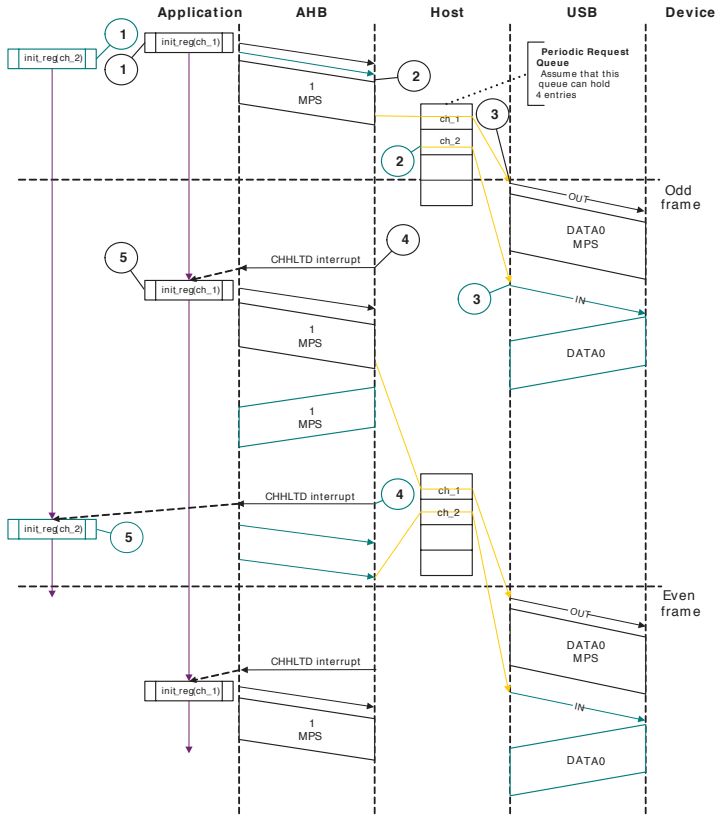


Figure 337: Normal Isochronous OUT/IN Transactions in DMA Mode

Interrupt Service Routine for Isochronous OUT Transactions in DMA Mode

Isochronous OUT

```
Unmask (CHHLTD)
if (CHHLTD)
{
    if (XFERCOMPL or FRMOVRUN)
    {
        De-allocate Channel
    }
}
```

Isochronous IN Transactions in DMA Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the

Core. Before it can communicate with the connected device, it must initialize a channel as described in Channel Initialization.

A typical isochronous IN operation in DMA mode is shown in Figure 337. 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.

Normal Isochronous IN Operation The sequence of operations in Figure 337 (channel 2) is as follows:

1. Initialize and enable channel 2 as explained in 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.

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)
        }
    }
}
```

AG.4.4 Device Programming Model

Before you program the Device, be sure to read Overview: Programming the Core and Modes of operation

Endpoint Initialization

This section addresses the following topics:

- Initialization on USB Reset
- Initialization on Enumeration Completion
- Initialization on SetAddress Command
- Initialization on SetConfiguration/SetInterface Command
- Endpoint Activation
- Endpoint Deactivation
- Device DMA/Slave Mode Initialization

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 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. 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).
   - 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.

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.

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.

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.

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 Endpoint Activation and 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 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.

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.

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.

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.

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).

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.

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 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.

Device Programming Operations

Table **??** provides links to the programming sequence for different USB transaction types.

Device Mode	IN	SETUP	OUT
Control			
Slave	Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode	OUT Data Transfers in Slave and DMA Modes	Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes
DMA	Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode	OUT Data Transfers in Slave and DMA Modes	Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes

Device Mode	IN	SETUP	OUT
Bulk			
Slave	Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode		Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes
DMA	Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode		Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes

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.

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 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in Endpoint Initialization. See 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

Device Mode	IN	SETUP	OUT
Interrupt			
Slave	Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding and Generic Periodic IN Data Transfers Without Thresholding Using the Periodic Transfer Interrupt Feature		Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes and Generic Interrupt OUT Data Transfers Without Thresholding Using Periodic Transfer Interrupt Feature
DMA	Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding and Generic Periodic IN Data Transfers Without Thresholding Using the Periodic Transfer Interrupt Feature		Generic Non-Isochronous OUT Data Transfers Without Thresholding in DMA and Slave Modes and Generic Interrupt OUT Data Transfers Without Thresholding Using Periodic Transfer Interrupt Feature

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.

Device Mode	IN	SETUP	OUT
Isochronous			
Slave	Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding		Control Read Transfers (SETUP, Data IN, Status OUT) and Incomplete Isochronous OUT Data Transfers in DMA and Slave Modes
DMA	Generic Periodic IN (Interrupt and Isochronous) Data Transfers Without Thresholding and Generic Periodic IN Data Transfers Without Thresholding Using the Periodic Transfer Interrupt Feature		Control Read Transfers (SETUP, Data IN, Status OUT) and Incomplete Isochronous OUT Data Transfers in DMA and Slave Modes

- 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 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 OTG 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 338 charts this flow.

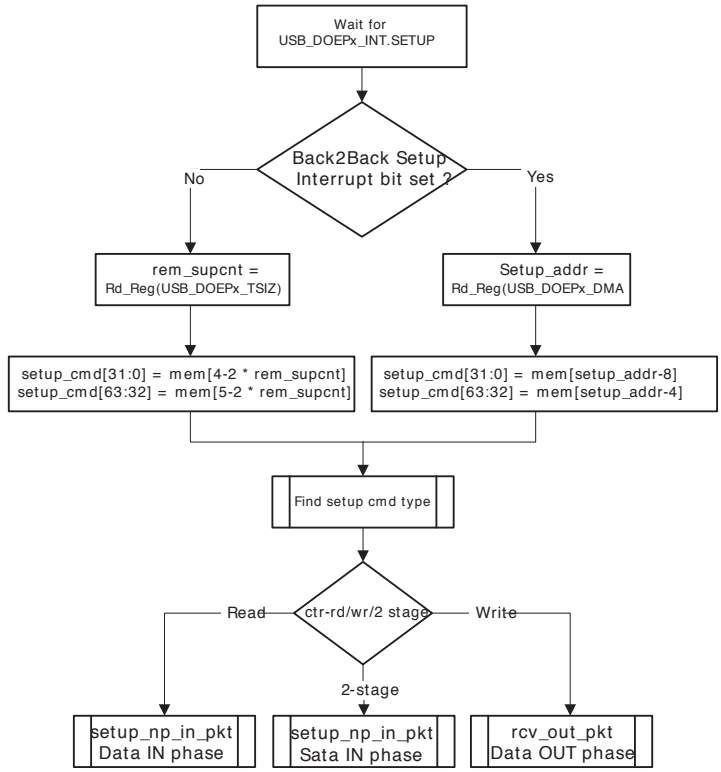


Figure 338: Processing a SETUP Packet

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.

Control Transfers This section describes the various types of control transfers.

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 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 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 Completed 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 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 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.

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 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 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 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 OUT Data Transfers in Slave and DMA Modes.
- USB_DOEPx_CTL.EPENA = 1

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 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 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 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 339.

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.

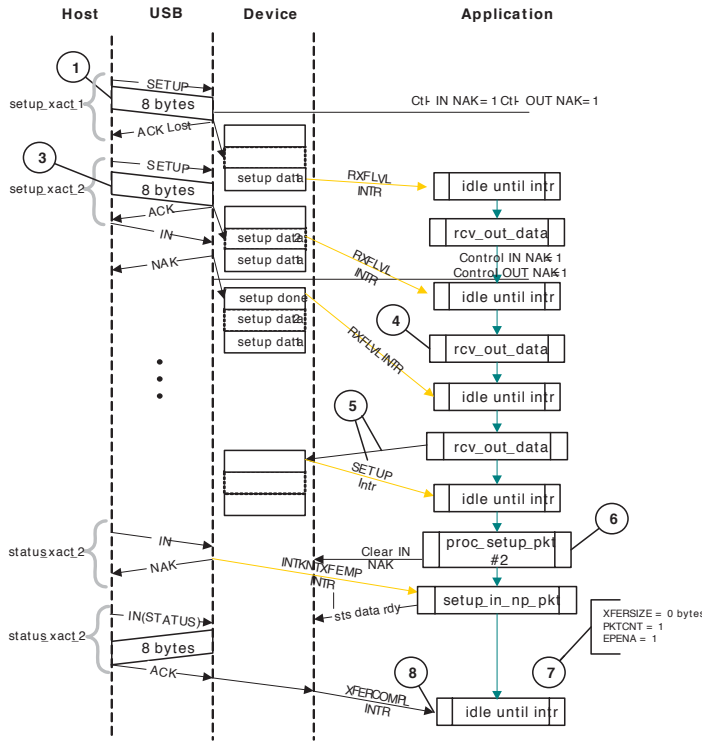


Figure 339: Two-Stage Control Transfer

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.
    1. 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.
    2. 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.
    3. 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.
    4. 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 Section AG.6.
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 340 provides a flow chart of this procedure.

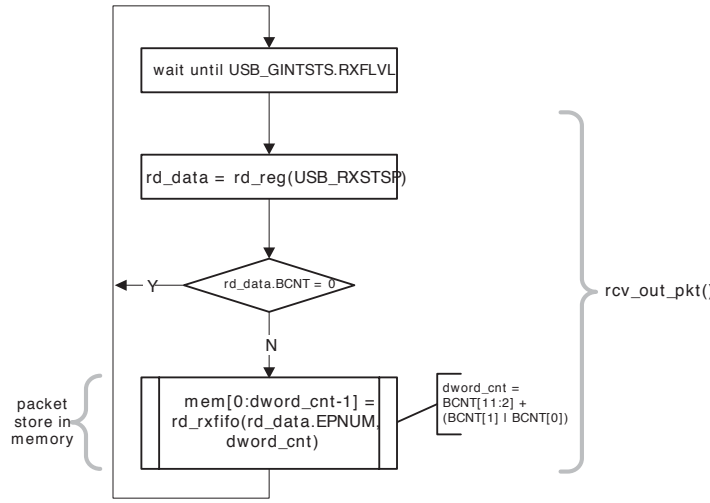


Figure 340: Receive FIFO Packet Read in Slave Mode

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 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.GOUTNAKE 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

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 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

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 Setting the Global OUT NAK.
2. Disable the required endpoint, as described in Section AG.4.4.
   - 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.

Generic Non-Isochronous OUT Data Transfers in DMA and Slave Modes To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in Endpoint Initialization. See 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 endpoint 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 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 341 depicts the reception of a single bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

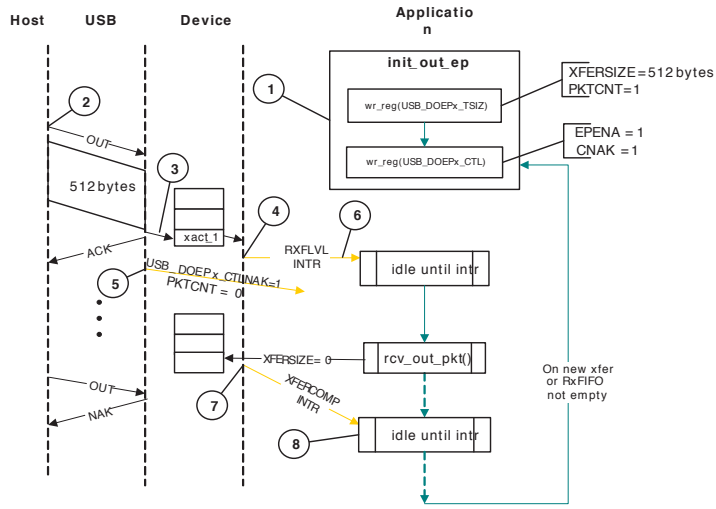


Figure 341: Slave Mode Bulk OUT Transaction

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.XFERCOMP interrupt bit to determine that the intended transfer is complete.

Generic Isochronous OUT Data Transfer in DMA and Slave Modes To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in Endpoint Initialization. See 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 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 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.

Generic Interrupt OUT Data Transfers 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 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in Endpoint Initialization. See 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.

Generic Isochronous OUT Data Transfers 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 Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in Endpoint Initialization. For packet writes in Slave mode, see: 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 342 gives the application flow for Isochronous OUT Periodic Transfer Interrupt feature.

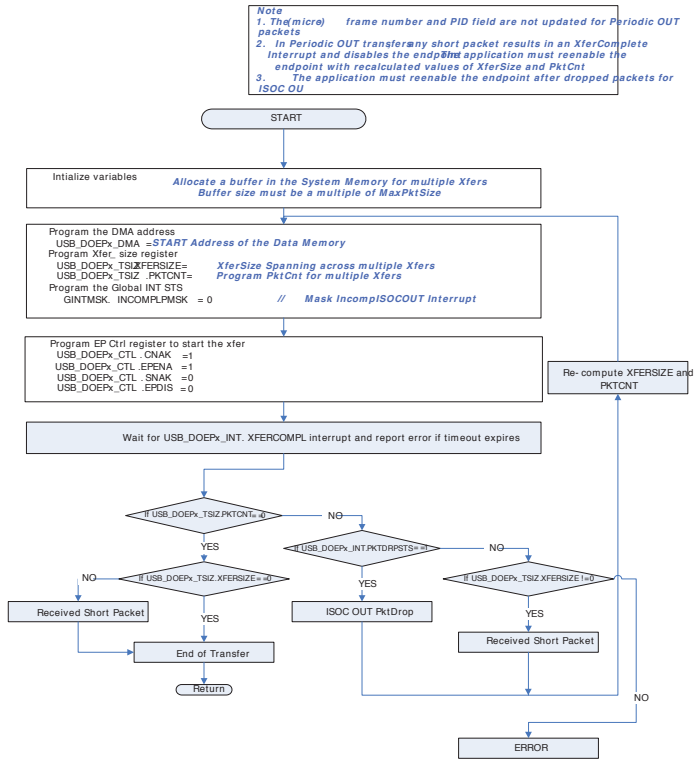


Figure 342: ISOC OUT Application Flow for Periodic Transfer Interrupt Feature

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.

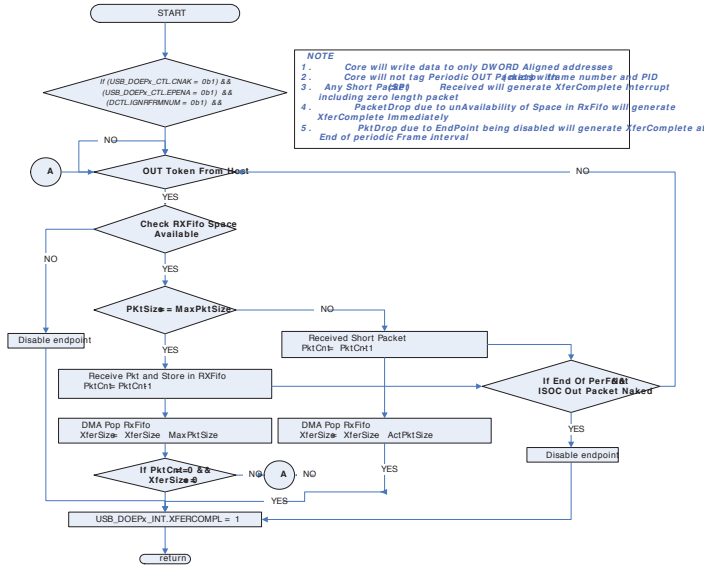


Figure 343: Isochronous OUT Core Internal Flow for Periodic Transfer Interrupt Feature

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 Overview: Programming the Core. Before it can communicate with the host, it

must initialize an endpoint as described in Endpoint Initialization. See 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.
   - 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 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 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.

IN Data Transfers in Slave and DMA Modes This section describes the internal data flow and application-level operations during IN data transfers.

- Packet Write in Slave Mode
- Setting Global Non-Periodic IN Endpoint NAK
- Setting IN Endpoint NAK
- IN Endpoint Disable
- Bulk IN Stall
- Incomplete Isochronous IN Data Transfers
- Stalling Non-Isochronous IN Endpoints
- Worst-Case Response Time
- Choosing the Value of USB_GUSBCFG.USBTRDTIM
- Handling Babble Conditions
- Generic Non-Periodic (Bulk and Control) IN Data Transfers Without Thresholding in DMA and Slave Mode
- Examples
-
- Generic Periodic IN Data Transfers Without Thresholding Using the Periodic Transfer Interrupt Feature

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.

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

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.INEPNAKEF (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

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 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.

Bulk IN Stall These notes refer to Figure 344

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

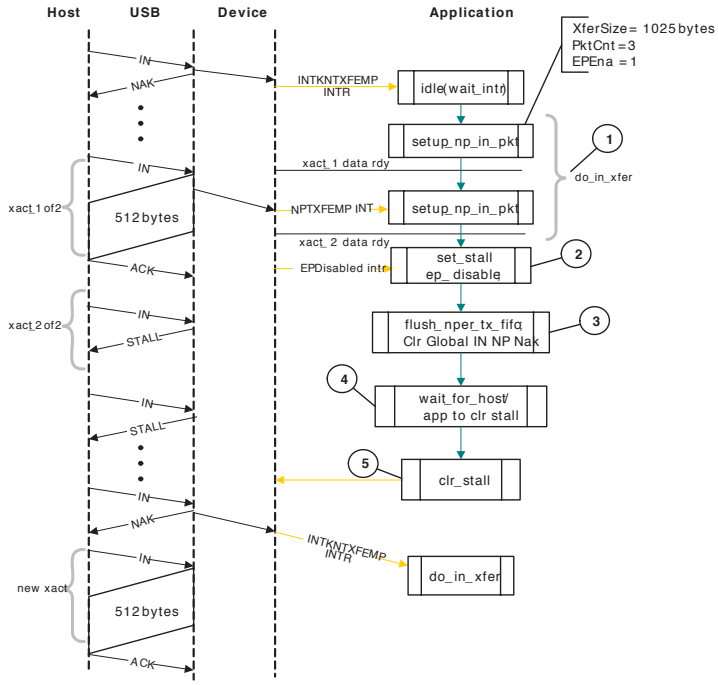


Figure 344: Bulk IN Stall

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.
   1. 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.

2. 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.
   1. The core transmits a zero-length data packet on the USB in response to the received IN token.

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.

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.

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.

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.

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 345 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)

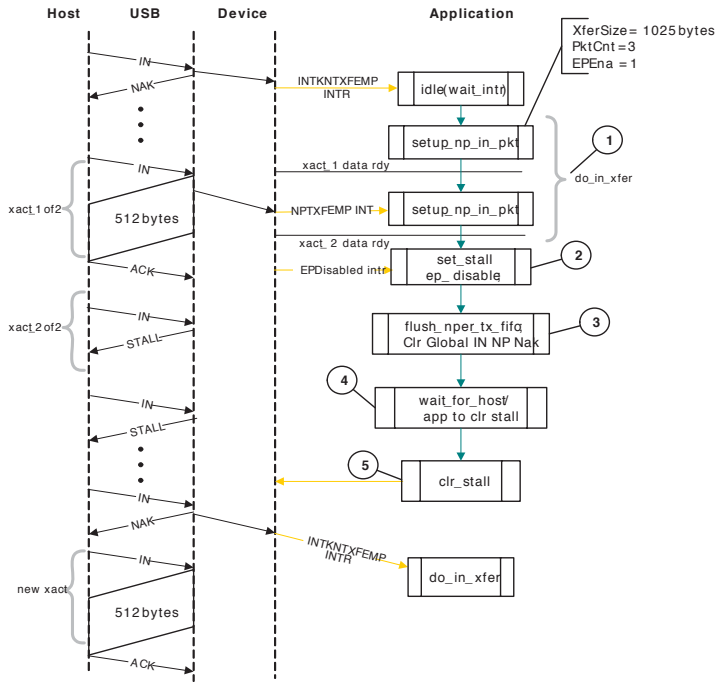


Figure 345: USBTRDTIM Max Timing Case ERROR wrong image

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.

Generic Non-Periodic (Bulk and Control) IN Data Transfers in DMA and Slave Mode To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in Endpoint Initialization. For packet writes in Slave mode, see: 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

   1. To transmit a single zero-length data packet:
      - Transfer size[epnum] = 0
      - Packet count[epnum] = 1
   2. 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.
   3. First transfer: transfer size[epnum] = n * mps[epnum]; packet count = n;
   4. 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.

Examples Slave Mode Bulk IN Transaction

These notes refer to Figure 346.

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.XFERCOMPL 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.

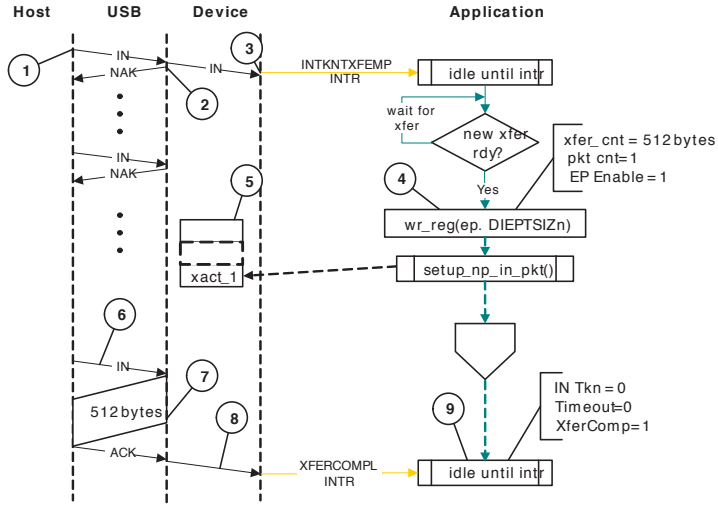


Figure 346: Slave Mode Bulk IN Transaction

Slave Mode Bulk IN Transfer (Pipelined Transaction)

These notes refer to Figure 347

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.INTKNTXFEM (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

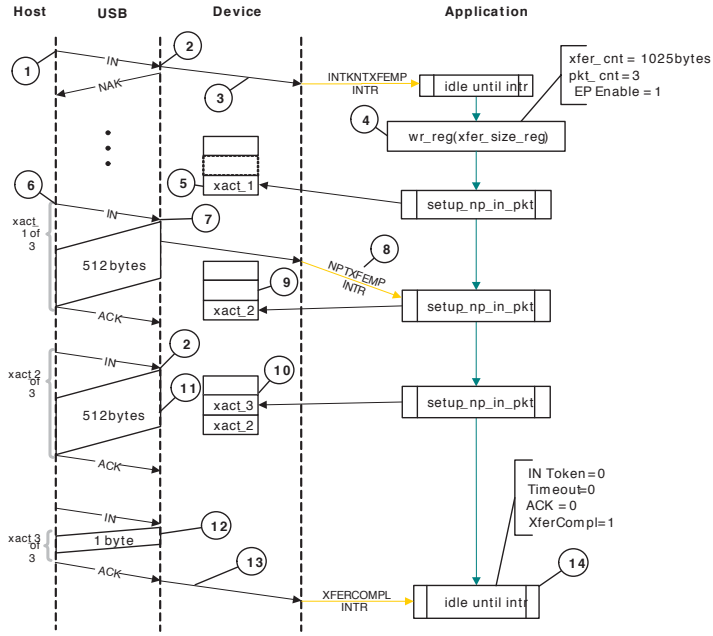


Figure 347: Slave Mode Bulk IN Transfer (Pipelined Transaction)

Slave Mode Bulk IN Two-Endpoint Transfer

These notes refer to Figure 348

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.

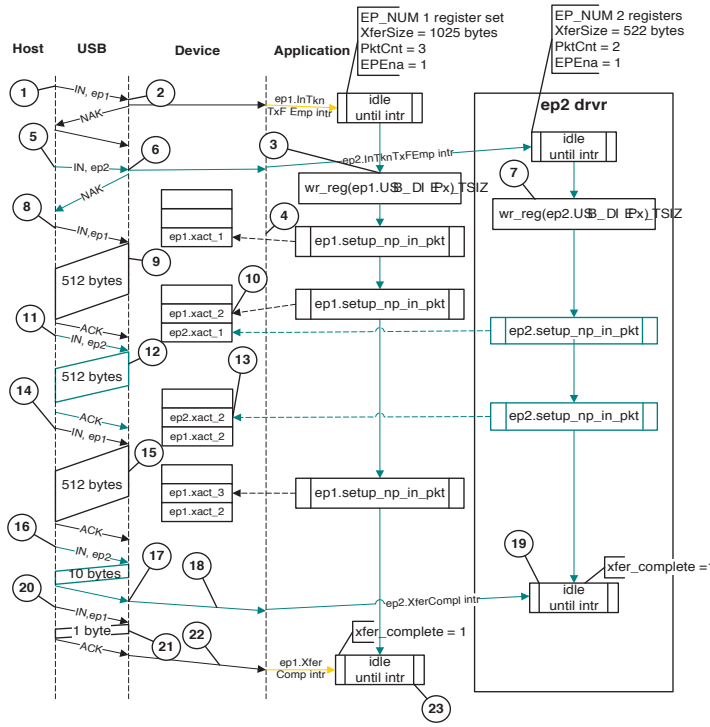


Figure 348: Slave Mode Bulk IN Two-Endpoint Transfer

Generic Periodic IN (Interrupt and Isochronous) Data Transfers To initialize the core after power-on reset, the application must follow the sequence in Overview: Programming the Core. Before it can communicate with the host, it must initialize an endpoint as described in Endpoint Initialization. For packet writes in Slave mode, see: Packet Write in Slave Mode.

Application Requirements

1. Application requirements 1, 2, 3, and 4 of 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 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.INTKN (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 Incomplete Isochronous IN Data Transfers, for more details.

Generic Periodic IN Data Transfers Using the Periodic Transfer Interrupt Feature This section describes a typical Periodic IN (ISOC / INTR) data transfer with the Periodic Transfer Interrupt feature.

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.
   1. 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
   2. To transmit a single zero-length data packet:
      - Transfer size[epnum] = 0
      - Packet count[epnum] = 1
   3. 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;

4. 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.
5. 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.

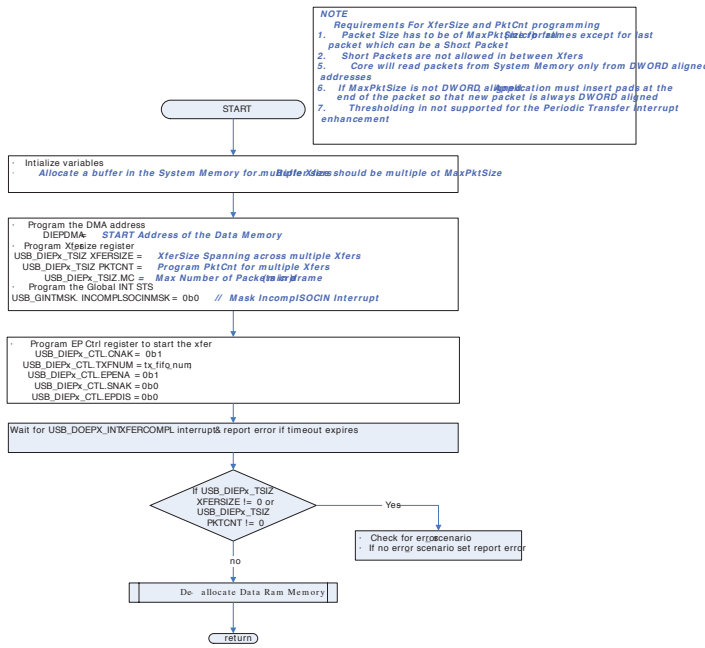


Figure 349: Periodic IN Application Flow for Periodic Transfer Interrupt Feature

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.

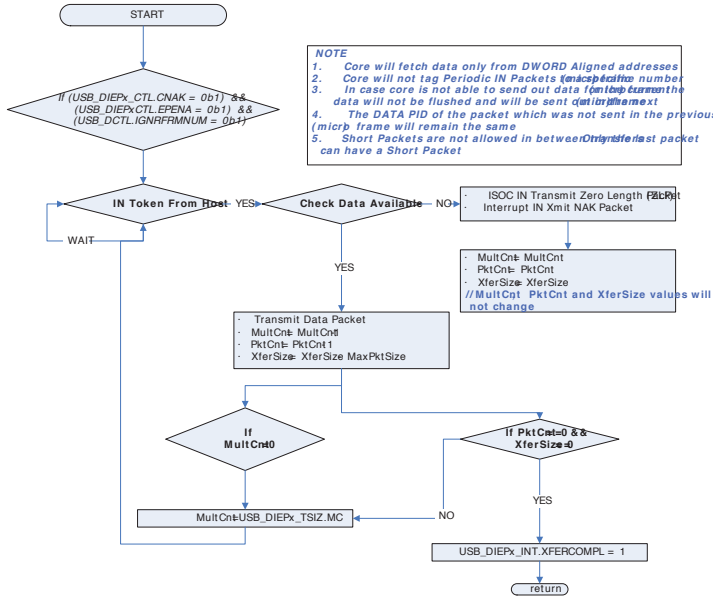


Figure 350: Periodic IN Core Internal Flow for Periodic Transfer Interrupt Feature

AG.4.5 OTG Revision 1.3 Programming Model

This section describes the OTG programming model when the core is configured to support OTG Revision 1.3 of the specification.

The core is an OTG device supporting HNP and SRP. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device. In Host mode, the core turns off Vbus to conserve power. SRP is a method by which the B-device signals the A-device to turn on Vbus power. A device must perform both data-line pulsing and Vbus pulsing, but a host can detect either data-line pulsing or Vbus pulsing for SRP. HNP is a

method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

A-Device Session Request Protocol

The application must set the SRP-Capable bit in the Core USB Configuration register. This enables the core to detect SRP as an A-device.

1. To save power, the application suspends and turns off port power when the bus is idle by writing the Port Suspend and Port Power bits in the Host Port Control and Status register.
2. PHY indicates port power off by detecting that VBUS voltage level is no longer valid.
3. The device must detect SE0 for at least 2 ms to start SRP when Vbus power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The core detects data-line pulsing.
5. The device drives Vbus above the A-device session valid (2.0 V minimum) for Vbus pulsing.

   The core interrupts the application on detecting SRP. The Session Request Detected bit is set in Global Interrupt Status register (USB_GINTSTS.SESSREQINT).
6. The application must service the Session Request Detected interrupt and turn on the Port Power bit by writing the Port Power bit in the Host Port Control and Status register. The PHY indicates port power-on by detecting a valid VBUS level.
7. When the USB is powered, the device connects, completing the SRP process.

B-Device Session Request Protocol

The application must set the SRP-Capable bit in the Core USB Configuration register. This enables the core to initiate SRP as a B-device. SRP is a means by which the core can request a new session from the host.

1. To save power, the host suspends and turns off port power when the bus is idle. PHY indicates port power off by detecting a not valid VBUS level.

   The core sets the Early Suspend bit in the Core Interrupt register after 3 ms of bus idleness. Following this, the core sets the USB Suspend bit in the Core Interrupt register.

   The PHY indicates the end of the B-device session by detecting a VBUS level below session valid.
2. PHY to enables the VBUS discharge function to speed up Vbus discharge.
3. The PHY indicates the session's end by detecting a session end voltage level on VBUS. This is the initial condition for SRP. The core requires 2 ms of SE0 before initiating SRP.

The application must wait until Vbus discharges to 0.2 V after USB_GOTGCTL.BSESVLD is deasserted. This discharge time can be obtained from the datasheet.

4. The application initiates SRP by writing the Session Request bit in the OTG Control and Status register. The core perform data-line pulsing followed by Vbus pulsing.

5. The host detects SRP from either the data-line or Vbus pulsing, and turns on Vbus. The PHY indicates Vbus power-on by detecting a valid VBUS level.

6. The core performs Vbus pulsing.

   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. The application reads the Session Request Success bit in the OTG Control and Status register.

7. When the USB is powered, the core connects, completing the SRP process.

A-Device Host Negotiation Protocol

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP-Capable bit in the Core USB Configuration register to enable the core to perform HNP as an A‑device.

1. The core sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set Host Set HNP Enable bit in the OTG Control and Status register to indicate to the core that the B-device supports HNP.

2. When it has finished using the bus, the application suspends by writing the Port Suspend bit in the Host Port Control and Status register.

3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

   The core sets the Host Negotiation Detected interrupt in the OTG Interrupt Status register, indicating the start of HNP.

   The PHY turns off the D+ and D- pulldown resistors to indicate a device role. The PHY enable the D+ pull-up resistor indicates a connect for B-device.

   The application must read the Current Mode bit in the OTG Control and Status register to determine Device mode operation.

4. The B-device detects the connection, issues a USB reset, and enumerates the core for data traffic.

5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

   The core sets the Early Suspend bit in the Core Interrupt register after 3 ms of bus idleness. Following this, the core sets the USB Suspend bit in the Core Interrupt register.

6. In Negotiated mode, the core detects the suspend, disconnects, and switches back to the host role. The core turns on the D+ and D- pulldown resistors to indicate its assumption of the host role.

7. The core sets the Connector ID Status Change interrupt in the OTG Interrupt Status register. The application must read the connector ID status in the OTG Control and Status register to determine the core's operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current Mode bit in the OTG Control and Status register to determine Host mode operation.

8. The B-device connects, completing the HNP process.

B-Device Host Negotiation Protocol

HNP switches the USB host role from B-device to A-device. The application must set the HNP-Capable bit in the Core USB Configuration register to enable the core to perform HNP as a B-device.

1. The A-device sends the SetFeature b_hnp_enable descriptor to enable HNP support. The core's ACK response indicates that it supports HNP. The application must set the Device HNP Enable bit in the OTG Control and Status register to indicate HNP support.

   The application sets the HNP Request bit in the OTG Control and Status register to indicate to the core to initiate HNP.

2. When it has finished using the bus, the A-device suspends by writing the Port Suspend bit in the Host Port Control and Status register.

   The core sets the Early Suspend bit in the Core Interrupt register after 3 ms of bus idleness. Following this, the core sets the USB Suspend bit in the Core Interrupt register.

   The core disconnects and the A-device detects SE0 on the bus, indicating HNP. The core enables the D+ and D- pulldown resistors to indicate its assumption of the host role.

   The A-device responds by activating its D+ pull-up resistor within 3 ms of detecting SE0. The core detects this as a connect.

   The core sets the Host Negotiation Success Status Change interrupt in the OTG Interrupt Status register, indicating the HNP status. The application must read the Host Negotiation Success bit in the OTG Control and Status register to determine host negotiation success. The application must read the Current Mode bit in the Core Interrupt register (USB_GINTSTS) to determine Host mode operation.

3. The application sets the reset bit (USB_HPRT.PRTRST) and the core issues a USB reset and enumerates the A-device for data traffic

4. The core continues the host role of initiating traffic, and when done, suspends the bus by writing the Port Suspend bit in the Host Port Control and Status register.

5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The core disables the D+ and D- pulldown resistors to indicate the assumption of the device role.

6. The application must read the Current Mode bit in the Core Interrupt (USB_GINTSTS) register to determine the Host mode operation.

7. The core connects, completing the HNP process.

AG.4.6 OTG Revision 2.0 Programming Model

OTG Revision 2.0 supports the new Attach Detection Protocol (ADP). This protocol enables a local device (an OTG device or Embedded Host) to detect when a remote device is attached or detached. **Note**

ADP is not supported by the core.

In addition to ADP, OTG Revision 2.0 also supports enhanced SRP and HNP, which are described in the following sections:

- OTG Revision 2.0 Session Request Protocol
- OTG Revision 2.0 Host Negotiation Protocol

Note

VBUS pulsing is not supported in OTG Revision 2.0 mode.

OTG Revision 2.0 Session Request Protocol

When the core is behaving as an A-device, it can power off VBUS when no session is active until the B-device initiates a SRP. The SRP detection is handled by the core.

Figure 351 illustrates the programming steps that need to be performed by A-device's application (core as A-device) when B-device initiates a SRP to establish a connection.

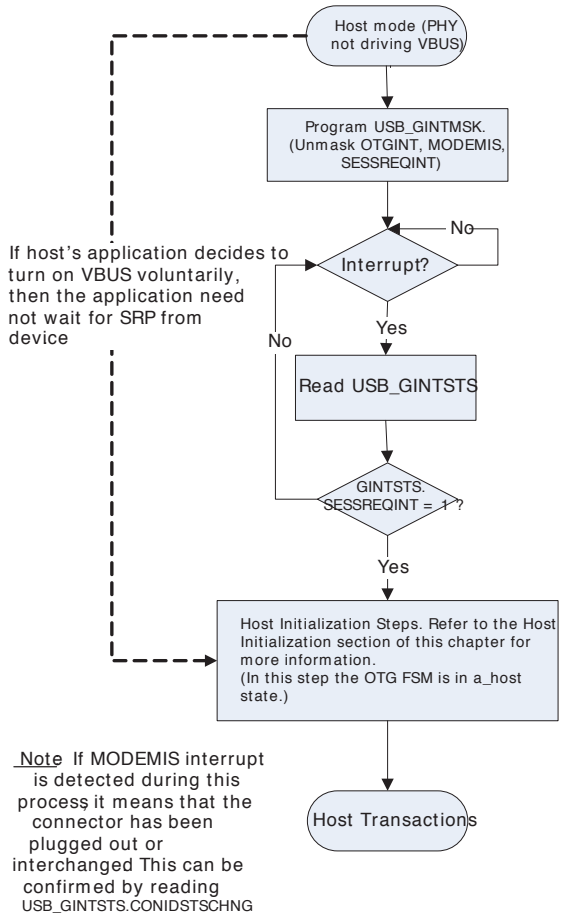


Figure 351: SRP Detection by Core When Operating as A-device

Figure 352 illustrates the steps that need to be performed by B-device's application (core as B-device) in order to establishing a connection with A-device by signaling a SRP.

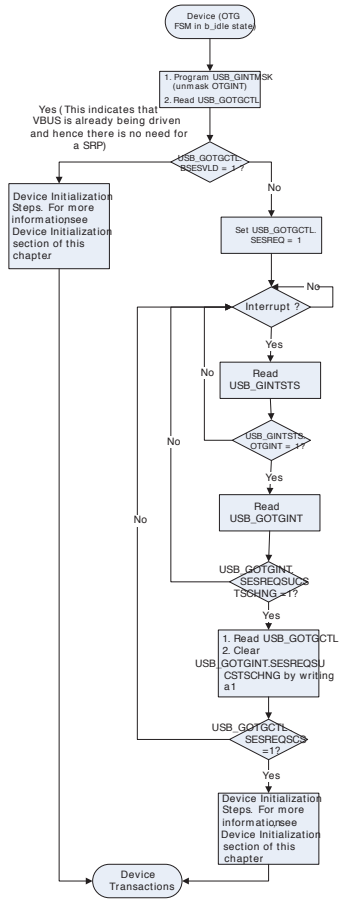


Figure 352: SRP Initiation by the Core When Acting as a B-Device

Note

The programming flow illustrated in Figure 352 is similar to OTG revision 1.3. This is because the presence or absence of VBUS pulsing is transparent to the application.

OTG Revision 2.0 Host Negotiation Protocol

When the core is operating as A-device, the application must execute a GetStatus() operation to the B-device with a frequency of THOST_REQ_POLL to determine the state of the host request flag in the B-device. If the host request flag is set in B-device it must program the core to change its role within THOST_REQ_SUSP.

Figure 353 shows the programming steps that need to be performed by A-device's application (core as A-device) in order to change its role to device. In Figure 353,

the A-device performs a role change, becomes a B-device and then reverts back to host (A-device) mode of operation.

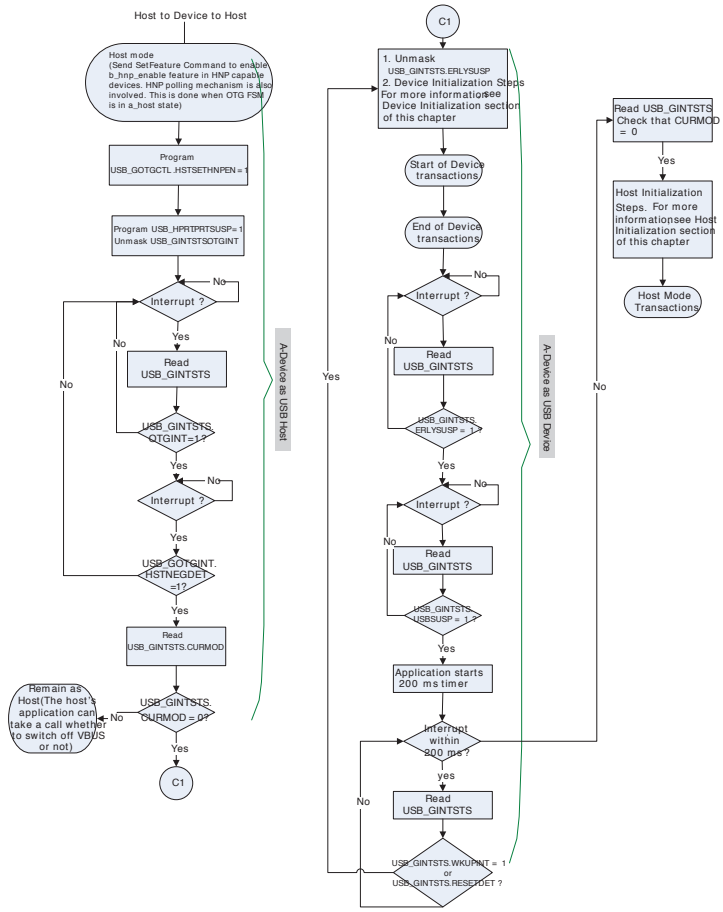


Figure 353: HNP When the Core is an A-Device

Figure 354 shows the programming steps that need to be performed by B-device's application (core as B-device) in order to change its role to Host. In Figure 354 , the B-device performs a role change, becomes a Host and then reverts back to Device mode of operation.

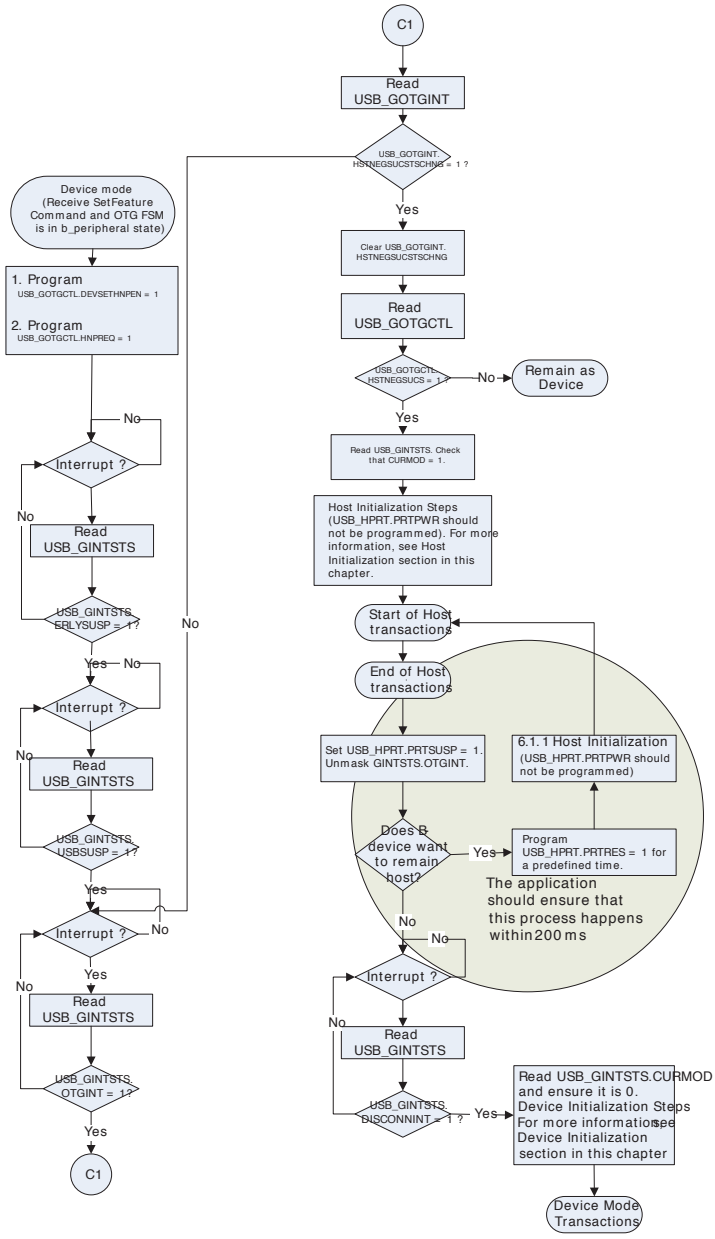


Figure 354: HNP When the Core is a B-Device

Note

During HNP process where the B-device is going to assume the role of a host, the B-device application needs to ensure that a USB reset process is programmed (in USB_HPRT register) within 150 ms (TB_ACON_BSE0) of getting a USB_HPRT.PRTCONNDET interrupt.

AG.4.7 FIFO RAM Allocation

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.

Device Mode

Tx FIFO Operation When allocating data RAM for FIFOs in Device mode keep in mind these factors:

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.

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.

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.

Summary of Guidelines for Choosing Data FIFO RAM Depth in Host Mode

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.

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).

Periodic TX FIFO size The recommended size for Periodic TXFIFO is sum total of (MPS*MC)/4 for all the channels. **Note**

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.

Calculating the Total FIFO Size 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)
```

Dynamic FIFO Allocation

The application can change the RAM allocation for each FIFO during the operation of the core.

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 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 Flushing TxFIFOs in the Core and Flushing RxFIFOs in the Core.

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 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 Flushing TxFIFOs in the Core and Flushing RxFIFOs in the Core.

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.

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.

The Core Interrupt Handler

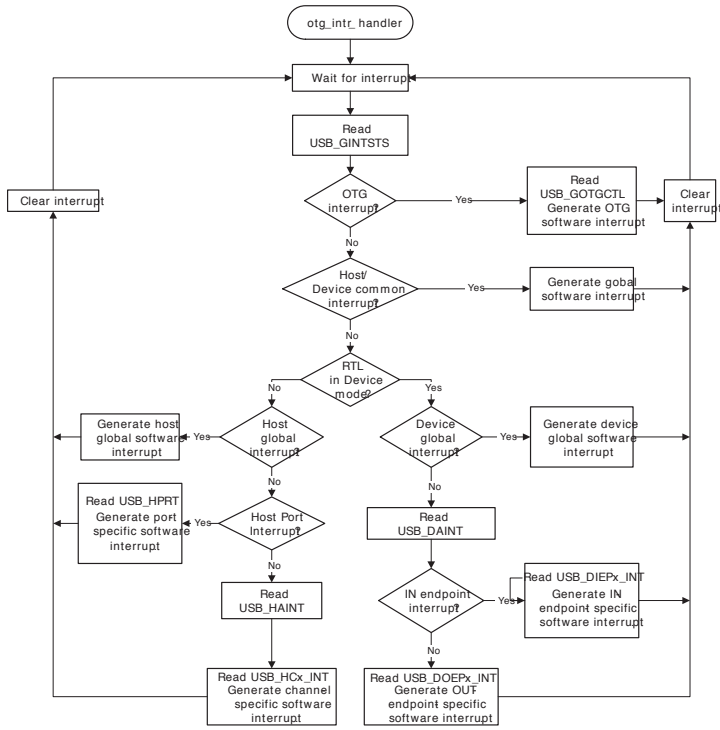


Figure 355: Core Interrupt Handler

AG.4.8 Suspend/Resume and SRP

This chapter describes different methods of saving power when the USB is suspended. This chapter discusses the following topics:

- Placing PHY in Low Power Mode Without Entering Suspend
  - When the Core is in Host Mode
  - When the Core is in Device Mode
- Suspend
  - Using EM2
    - Overview of the EM2 Programming Model
    - Using EM2 when the Core is in Host Mode
    - EM2 when the Core is in Device Mode
- Clock Gating (EM0 and EM1)

- Internal Clock Gating when the Core is in Host Mode
- Internal Clock Gating when the Core is in Device Mode

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.

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.

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.

Suspend

When the core is in Suspend, the following power conservation options are available to use:

- Using EM2: You can enter EM2, turning off power (and reseting) parts of the core
- Clock Gating (EM0 and EM1): You can choose gate the AHB clock to some parts of the core Internal Clock Gating when the Core is in Host Mode

This section discusses methods of conserving power by using one of the above methods.

Using EM2

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.

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.

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 USB Core Clock (USBC) 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 USB Core Clock (USBC) 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 USB Core Clock (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 USB Core clock (USBC) 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 USB Core clock (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.

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.

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.

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.

AG.4.9 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 [0x3C000, 0x3C3FF] and typically start with first letter G.
- Host Mode Registers are located in the address offset-range [0x3C400, 0x3C7FF] and start with first letter H.
- Device Mode Registers are located in the address offset-range [0x3C800, 0x3CDFF] and start with first letter D.
- The Power and Clock Gating register is located at offset 0x3CE00.

- The Device EP/Host Channel FIFOs start at address offset 0x3D000 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 0x5C000.

AG.5 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	W1	Interrupt Flag Clear Register
0x014	USB_IEN	RW	Interrupt Enable Register
0x018	USB_ROUTE	RW	I/O Routing Register
0x3C000	USB_GOTGCTL	RWH	OTG Control and Status Register
0x3C004	USB_GOTGINT	RW1H	OTG Interrupt Register
0x3C008	USB_GAHBCFG	RW	AHB Configuration Register
0x3C00C	USB_GUSBCFG	RWH	USB Configuration Register
0x3C010	USB_GRSTCTL	RWH	Reset Register
0x3C014	USB_GINTSTS	RWH	Interrupt Register
0x3C018	USB_GINTMSK	RW	Interrupt Mask Register
0x3C01C	USB_GRXSTSR	R	Receive Status Debug Read Register
0x3C020	USB_GRXSTSP	R	Receive Status Read and Pop Register
0x3C024	USB_GRXFSIZ	RW	Receive FIFO Size Register
0x3C028	USB_GNPTXFSIZ	RW	Non-periodic Transmit FIFO Size Register
0x3C02C	USB_GNPTXSTS	R	Non-periodic Transmit FIFO/Queue Status Register
0x3C05C	USB_GDFIFOCFG	RW	Global DFIFO Configuration Register
0x3C100	USB_HPTXFSIZ	RW	Host Periodic Transmit FIFO Size Register
0x3C104	USB_DIEPTXF1	RW	Device IN Endpoint Transmit FIFO 1 Size Register
0x3C108	USB_DIEPTXF2	RW	Device IN Endpoint Transmit FIFO 2 Size Register
0x3C10C	USB_DIEPTXF3	RW	Device IN Endpoint Transmit FIFO 3 Size Register

Offset	Name	Type	Description
0x3C110	USB_DIEPTXF4	RW	Device IN Endpoint Transmit FIFO 4 Size Register
0x3C114	USB_DIEPTXF5	RW	Device IN Endpoint Transmit FIFO 5 Size Register
0x3C118	USB_DIEPTXF6	RW	Device IN Endpoint Transmit FIFO 6 Size Register
0x3C400	USB_HCFG	RW	Host Configuration Register
0x3C404	USB_HFIR	RW	Host Frame Interval Register
0x3C408	USB_HFNUM	R	Host Frame Number/Frame Time Remaining Register
0x3C410	USB_HPTXSTS	R	Host Periodic Transmit FIFO/Queue Status Register
0x3C414	USB_HAINT	R	Host All Channels Interrupt Register
0x3C418	USB_HAINTMSK	RW	Host All Channels Interrupt Mask Register
0x3C440	USB_HPRT	RWH	Host Port Control and Status Register
0x3C500	USB_HC0_CHAR	RWH	Host Channel x Characteristics Register
0x3C508	USB_HC0_INT	RW1H	Host Channel x Interrupt Register
0x3C50C	USB_HC0_INTMSK	RW	Host Channel x Interrupt Mask Register
0x3C510	USB_HC0_TSIZ	RW	Host Channel x Transfer Size Register
0x3C514	USB_HC0_DMAADDR	RW	Host Channel x DMA Address Register
...	USB_HCx_CHAR	RWH	Host Channel x Characteristics Register
...	USB_HCx_INT	RW1H	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
...	USB_HCx_DMAADDR	RW	Host Channel x DMA Address Register
0x3C6A0	USB_HC13_CHAR	RWH	Host Channel x Characteristics Register
0x3C6A8	USB_HC13_INT	RW1H	Host Channel x Interrupt Register

Offset	Name	Type	Description
0x3C6AC	USB_HC13_INTMSK	RW	Host Channel x Interrupt Mask Register
0x3C6B0	USB_HC13_TSIZ	RW	Host Channel x Transfer Size Register
0x3C6B4	USB_HC13_DMAADDR	RW	Host Channel x DMA Address Register
0x3C800	USB_DCFG	RW	Device Configuration Register
0x3C804	USB_DCTL	RWH	Device Control Register
0x3C808	USB_DSTS	R	Device Status Register
0x3C810	USB_DIEPMSK	RW	Device IN Endpoint Common Interrupt Mask Register
0x3C814	USB_DOEPMSK	RW	Device OUT Endpoint Common Interrupt Mask Register
0x3C818	USB_DAINT	R	Device All Endpoints Interrupt Register
0x3C81C	USB_DAINTMSK	RW	Device All Endpoints Interrupt Mask Register
0x3C828	USB_DVBUSDIS	RW	Device VBUS Discharge Time Register
0x3C82C	USB_DVBUSPULSE	RW	Device VBUS Pulsing Time Register
0x3C834	USB_DIEPEMPMSK	RW	Device IN Endpoint FIFO Empty Interrupt Mask Register
0x3C900	USB_DIEP0CTL	RWH	Device IN Endpoint 0 Control Register
0x3C908	USB_DIEP0INT	RWH	Device IN Endpoint 0 Interrupt Register
0x3C910	USB_DIEP0TSIZ	RW	Device IN Endpoint 0 Transfer Size Register
0x3C914	USB_DIEP0DMAADDR	RW	Device IN Endpoint 0 DMA Address Register
0x3C918	USB_DIEP0TXFSTS	R	Device IN Endpoint 0 Transmit FIFO Status Register
0x3C920	USB_DIEP0_CTL	RWH	Device IN Endpoint x+1 Control Register
0x3C928	USB_DIEP0_INT	RWH	Device IN Endpoint x+1 Interrupt Register
0x3C930	USB_DIEP0_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register

Offset	Name	Type	Description
0x3C934	USB_DIEP0_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
0x3C938	USB_DIEP0_TXFSTS	R	Device IN Endpoint x+1 Transmit FIFO Status Register
0x3C940	USB_DIEP1_CTL	RWH	Device IN Endpoint x+1 Control Register
0x3C948	USB_DIEP1_INT	RWH	Device IN Endpoint x+1 Interrupt Register
0x3C950	USB_DIEP1_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register
0x3C954	USB_DIEP1_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
0x3C958	USB_DIEP1_TXFSTS	R	Device IN Endpoint x+1 Transmit FIFO Status Register
0x3C960	USB_DIEP2_CTL	RWH	Device IN Endpoint x+1 Control Register
0x3C968	USB_DIEP2_INT	RWH	Device IN Endpoint x+1 Interrupt Register
0x3C970	USB_DIEP2_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register
0x3C974	USB_DIEP2_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
0x3C978	USB_DIEP2_TXFSTS	R	Device IN Endpoint x+1 Transmit FIFO Status Register
0x3C980	USB_DIEP3_CTL	RWH	Device IN Endpoint x+1 Control Register
0x3C988	USB_DIEP3_INT	RWH	Device IN Endpoint x+1 Interrupt Register
0x3C990	USB_DIEP3_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register
0x3C994	USB_DIEP3_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
0x3C998	USB_DIEP3_TXFSTS	R	Device IN Endpoint x+1 Transmit FIFO Status Register
0x3C9A0	USB_DIEP4_CTL	RWH	Device IN Endpoint x+1 Control Register
0x3C9A8	USB_DIEP4_INT	RWH	Device IN Endpoint x+1 Interrupt Register
0x3C9B0	USB_DIEP4_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register

Offset	Name	Type	Description
0x3C9B4	USB_DIEP4_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
0x3C9B8	USB_DIEP4_TXFSTS	R	Device IN Endpoint x+1 Transmit FIFO Status Register
0x3C9C0	USB_DIEP5_CTL	RWH	Device IN Endpoint x+1 Control Register
0x3C9C8	USB_DIEP5_INT	RWH	Device IN Endpoint x+1 Interrupt Register
0x3C9D0	USB_DIEP5_TSIZ	RW	Device IN Endpoint x+1 Transfer Size Register
0x3C9D4	USB_DIEP5_DMAADDR	RW	Device IN Endpoint x+1 DMA Address Register
0x3C9D8	USB_DIEP5_TXFSTS	R	Device IN Endpoint x+1 Transmit FIFO Status Register
0x3CB00	USB_DOEP0CTL	RWH	Device OUT Endpoint 0 Control Register
0x3CB08	USB_DOEP0INT	RW1H	Device OUT Endpoint 0 Interrupt Register
0x3CB10	USB_DOEP0TSIZ	RW	Device OUT Endpoint 0 Transfer Size Register
0x3CB14	USB_DOEP0DMAADDR	RW	Device OUT Endpoint 0 DMA Address Register
0x3CB20	USB_DOEP0_CTL	RWH	Device OUT Endpoint x+1 Control Register
0x3CB28	USB_DOEP0_INT	RW1H	Device OUT Endpoint x+1 Interrupt Register
0x3CB30	USB_DOEP0_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
0x3CB34	USB_DOEP0_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register
0x3CB40	USB_DOEP1_CTL	RWH	Device OUT Endpoint x+1 Control Register
0x3CB48	USB_DOEP1_INT	RW1H	Device OUT Endpoint x+1 Interrupt Register
0x3CB50	USB_DOEP1_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
0x3CB54	USB_DOEP1_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register
0x3CB60	USB_DOEP2_CTL	RWH	Device OUT Endpoint x+1 Control Register

Offset	Name	Type	Description
0x3CB68	USB_DOEP2_INT	RW1H	Device OUT Endpoint x+1 Interrupt Register
0x3CB70	USB_DOEP2_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
0x3CB74	USB_DOEP2_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register
0x3CB80	USB_DOEP3_CTL	RWH	Device OUT Endpoint x+1 Control Register
0x3CB88	USB_DOEP3_INT	RW1H	Device OUT Endpoint x+1 Interrupt Register
0x3CB90	USB_DOEP3_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
0x3CB94	USB_DOEP3_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register
0x3CBA0	USB_DOEP4_CTL	RWH	Device OUT Endpoint x+1 Control Register
0x3CBA8	USB_DOEP4_INT	RW1H	Device OUT Endpoint x+1 Interrupt Register
0x3CBB0	USB_DOEP4_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
0x3CBB4	USB_DOEP4_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register
0x3CBC0	USB_DOEP5_CTL	RWH	Device OUT Endpoint x+1 Control Register
0x3CBC8	USB_DOEP5_INT	RW1H	Device OUT Endpoint x+1 Interrupt Register
0x3CBD0	USB_DOEP5_TSIZ	RWH	Device OUT Endpoint x+1 Transfer Size Register
0x3CBD4	USB_DOEP5_DMAADDR	RW	Device OUT Endpoint x+1 DMA Address Register
0x3CE00	USB_PCGCCTL	RWH	Power and Clock Gating Control Register
0x3D000	USB_FIFO0D0	RW	Device EP 0/Host Channel 0 FIFO
...	USB_FIFO0Dx	RW	Device EP 0/Host Channel 0 FIFO
0x3D7FC	USB_FIFO0D511	RW	Device EP 0/Host Channel 0 FIFO
0x3E000	USB_FIFO1D0	RW	Device EP 1/Host Channel 1 FIFO
...	USB_FIFO1Dx	RW	Device EP 1/Host Channel 1 FIFO
0x3E7FC	USB_FIFO1D511	RW	Device EP 1/Host Channel 1 FIFO
0x3F000	USB_FIFO2D0	RW	Device EP 2/Host Channel 2 FIFO

Offset	Name	Type	Description
...	USB_FIFO2Dx	RW	Device EP 2/Host Channel 2 FIFO
0x3F7FC	USB_FIFO2D511	RW	Device EP 2/Host Channel 2 FIFO
0x40000	USB_FIFO3D0	RW	Device EP 3/Host Channel 3 FIFO
...	USB_FIFO3Dx	RW	Device EP 3/Host Channel 3 FIFO
0x407FC	USB_FIFO3D511	RW	Device EP 3/Host Channel 3 FIFO
0x41000	USB_FIFO4D0	RW	Device EP 4/Host Channel 4 FIFO
...	USB_FIFO4Dx	RW	Device EP 4/Host Channel 4 FIFO
0x417FC	USB_FIFO4D511	RW	Device EP 4/Host Channel 4 FIFO
0x42000	USB_FIFO5D0	RW	Device EP 5/Host Channel 5 FIFO
...	USB_FIFO5Dx	RW	Device EP 5/Host Channel 5 FIFO
0x427FC	USB_FIFO5D511	RW	Device EP 5/Host Channel 5 FIFO
0x43000	USB_FIFO6D0	RW	Device EP 6/Host Channel 6 FIFO
...	USB_FIFO6Dx	RW	Device EP 6/Host Channel 6 FIFO
0x437FC	USB_FIFO6D511	RW	Device EP 6/Host Channel 6 FIFO
0x44000	USB_FIFO7D0	RW	Host Channel 7 FIFO
...	USB_FIFO7Dx	RW	Host Channel 7 FIFO
0x447FC	USB_FIFO7D511	RW	Host Channel 7 FIFO
0x45000	USB_FIFO8D0	RW	Host Channel 8 FIFO
...	USB_FIFO8Dx	RW	Host Channel 8 FIFO
0x457FC	USB_FIFO8D511	RW	Host Channel 8 FIFO
0x46000	USB_FIFO9D0	RW	Host Channel 9 FIFO
...	USB_FIFO9Dx	RW	Host Channel 9 FIFO
0x467FC	USB_FIFO9D511	RW	Host Channel 9 FIFO
0x47000	USB_FIFO10D0	RW	Host Channel 10 FIFO
...	USB_FIFO10Dx	RW	Host Channel 10 FIFO
0x477FC	USB_FIFO10D511	RW	Host Channel 10 FIFO
0x48000	USB_FIFO11D0	RW	Host Channel 11 FIFO
...	USB_FIFO11Dx	RW	Host Channel 11 FIFO
0x487FC	USB_FIFO11D511	RW	Host Channel 11 FIFO
0x49000	USB_FIFO12D0	RW	Host Channel 12 FIFO
...	USB_FIFO12Dx	RW	Host Channel 12 FIFO
0x497FC	USB_FIFO12D511	RW	Host Channel 12 FIFO
0x4A000	USB_FIFO13D0	RW	Host Channel 13 FIFO

Offset	Name	Type	Description
...	USB_FIFO13Dx	RW	Host Channel 13 FIFO
0x4A7FC	USB_FIFO13D511	RW	Host Channel 13 FIFO
0x5C000	USB_FIFORAM0	RW	Direct Access to Data FIFO RAM for Debugging (2 KB)
...	USB_FIFORAMx	RW	Direct Access to Data FIFO RAM for Debugging (2 KB)
0x5C7FC	USB_FIFORAM511	RW	Direct Access to Data FIFO RAM for Debugging (2 KB)

AG.6 Register Description

AG.6.1 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							0x0				0x0				0	0															0	0
Access							RW				RW				RW	RW															RW	RW
Name							BIASPROGEM23				BIASPROGEM01				VREGOSEN	VREGDIS													DMPUAP	VBUSENAP

Bit	Name	Reset	Access	Description
31:26	Reserved	To ensure compatibility with future devices, always write bits to 0.		
25:24	BIASPROGEM23	0x0	RW	**Regulator Bias Programming Value in EM2/3**
	Regulator bias current setting in EM2/3 (i.e. while USB in suspend).			
23:22	Reserved	To ensure compatibility with future devices, always write bits to 0.		
21:20	BIASPROGEM01	0x0	RW	**Regulator Bias Programming Value in EM0/1**
	Regulator bias current setting in EM0/1 (i.e. while USB active).			
19:18	Reserved	To ensure compatibility with future devices, always write bits to 0.		
17	VREGOSEN	0	RW	**VREGO Sense Enable**
	Set this bit to enable USB_VREGO voltage level sensing.			
16	VREGDIS	0	RW	**Voltage Regulator Disable**
	Set this bit to disable the voltage regulator.			
15:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	DMPUAP	0	RW	**DMPU Active Polarity**
	Use this bit to select the active polarity of the USB_DMPU pin.			

Value	Mode	Description
0	LOW	USB_DMPU is active low.
1	HIGH	USB_DMPU is active high.

Bit	Name	Reset	Access	Description
0	VBUSENAP	0	RW	**VBUSEN Active Polarity**
	Use this bit to select the active polarity of the USB_VBUSEN pin.			

Value	Mode	Description
0	LOW	USB_VBUSEN is active low.
1	HIGH	USB_VBUSEN is active high.

AG.6.2 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
Access																																R
Name																																VREGOS

Bit	Name	Reset	Access	Description
31:1	Reserved	To ensure compatibility with future devices, always write bits to 0.		
0	VREGOS	0	R	**VREGO Sense Output**

Bit	Name	Reset	Access	Description
	USB_VREGO Voltage Sense output. 0 when no USB_VREGO voltage, 1 when USB_VREGO above approximately 1.8 V. Always 0 when VREGOSEN in USB_CTRL is 0.			

AG.6.3 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																															1	1
Access																															R	R
Name																															VREGOSL	VREGOSH

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	VREGOSL	1	R	**VREGO Sense Low Interrupt Flag**
	Set when USB_VREGO drops below approximately 1.8 V.			
0	VREGOSH	1	R	**VREGO Sense High Interrupt Flag**
	Set when USB_VREGO goes above approximately 1.8 V.			

AG.6.4 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
Access																															W1	W1
Name																															VREGOSL	VREGOSH

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	VREGOSL	0	W1	**Set VREGO Sense Low Interrupt Flag**
	Write to 1 to set the VREGO Sense Low Interrupt Flag.			
0	VREGOSH	0	W1	**Set VREGO Sense High Interrupt Flag**
	Write to 1 to set the VREGO Sense High Interrupt Flag.			

AG.6.5 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
Access																															W1	W1
Name																															VREGOSL	VREGOSH

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	VREGOSL	0	W1	**Clear VREGO Sense Low Interrupt Flag**
	Write to 1 to clear the VREGO Sense Low Interrupt Flag.			
0	VREGOSH	0	W1	**Clear VREGO Sense High Interrupt Flag**
	Write to 1 to clear the VREGO Sense High Interrupt Flag.			

AG.6.6 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
Access																															RW	RW
Name																															VREGOSL	VREGOSH

Bit	Name	Reset	Access	Description
31:2	Reserved	To ensure compatibility with future devices, always write bits to 0.		
1	VREGOSL	0	RW	**VREGO Sense Low Interrupt Enable**
	Enable interrupt on VREGO Sense Low.			
0	VREGOSH	0	RW	**VREGO Sense High Interrupt Enable**
	Enable interrupt on VREGO Sense High.			

AG.6.7 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	0
Access																														RW	RW	RW
Name																														DMPUPEN	VBUSENPEN	PHYPEN

Bit	Name	Reset	Access	Description
31:3	Reserved	To ensure compatibility with future devices, always write bits to 0.		
2	DMPUPEN	0	RW	**DMPU Pin Enable**
	When set, the USB_DMPU pin is enabled.			
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.			

AG.6.8 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																													
0x3C000	31	30	29	28	27	26	25	24	23	22	21	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	0	0	0	0
Access												RW	R	R	R	R					RW	RW	RW	R	RW	RW	RW	RW	RW	RW	R
Name												OTGVER	BSESVLD	ASESVLD	DBNCTIME	CONIDSTS					DEVHNPEN	HSTSETHNPEN	HNPREQ	HSTNEGSCS	AVALIDOVVAL	AVALIDOVEN	BVALIDOVVAL	BVALIDOVEN	VBVALIDOVVAL	VBVALIDOVEN	SESREQ	SESREQSCS

Bit	Name	Reset	Access	Description
31:21	Reserved	To ensure compatibility with future devices, always write bits to 0.		
20	OTGVER	0	RW	**OTG Version**

Indicates the OTG revision.

Value	Mode	Description
0	OTG13	OTG Version 1.3. In this version the core supports data line pulsing and VBus pulsing for SRP.
1	OTG20	OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

19	BSESVLD	0	R	**B-Session Valid (device only)**

Indicates the Device mode transceiver status for B-session valid. In OTG mode, you can use this bit to determine if the device is connected or disconnected.

18	ASESVLD	0	R	**A-Session Valid (host only)**

Indicates the Host mode transceiver status for A-session valid.

17	DBNCTIME	0	R	**Long/Short Debounce Time (host only)**

Indicates the debounce time of a detected connection.

Value	Mode	Description
0	LONG	Long debounce time, used for physical connections (100 ms + 2.5 us).
1	SHORT	Short debounce time, used for soft connections (2.5 us).

16	CONIDSTS	1	R	**Connector ID Status (host and device)**

Indicates the connector ID status on a connect event.

Value	Mode	Description
0	A	The core is in A-Device mode.
1	B	The core is in B-Device mode.

15:12	Reserved	To ensure compatibility with future devices, always write bits to 0.		
11	DEVHNPEN	0	RW	**Device HNP Enabled (device only)**

The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.

10	HSTSETHNPEN	0	RW	**Host Set HNP Enable (host only)**

The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.

9	HNPREQ	0	RW	**HNP Request (device only)**

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 (USB_GOTGINT.HSTNEGSUCSTSCHNG) is set. The core clears this bit when the HSTNEGSUCSTSCHNG bit is cleared.

8	HSTNEGSCS	0	R	**Host Negotiation Success (device only)**

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.

7	AVALIDOVVAL	0	RW	**Avalid Override Value**

This bit is used to set Override value for Avalid signal when USB_GOTGCTL.AVALIDOVEN is set.

6	AVALIDOVEN	0	RW	**AValid Override Enable**

This bit is used to enable/disable the software to override the Avalid signal using the USB_GOTGCTL.AVALIDOVVAL. When set Avalid received from the PHY is overridden with USB_GOTGCTL.AVALIDOVVAL.

5	BVALIDOVVAL	0	RW	**Bvalid Override Value**

This bit is used to set Override value for Bvalid signal when USB_GOTGCTL.BVALIDOVEN is set.

4	BVALIDOVEN	0	RW	**BValid Override Enable**

This bit is used to enable/disable the software to override the Bvalid signal using the USB_GOTGCTL.BVALIDOVVAL. When set Bvalid received from the PHY is overridden with USB_GOTGCTL.BVALIDOVVAL.

3	VBVALIDOVVAL	0	RW	**VBUS Valid Override Value**

This bit is used to set Override value for vbusvalid signal when USB_GOTGCTL.VBVALIDOVEN is set.

2	VBVALIDOVEN	0	RW	**VBUS-Valid Override Enable**

Bit	Name	Reset	Access	Description
	This bit is used to enable/disable the software to override the vbusvalid signal using the USB_GOTGCTL.VBVALIDOVVAL. When set, vbusvalid received from the PHY is overridden with USB_GOTGCTL.VBVALIDOVVAL.			
1	SESREQ	0	RW	**Session Request (device only)**
	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 (USB_GOTGINT.HSTNEGSUCSTSCHNG) is set. The core clears this bit when the HSTNEGSUCSTSCHNG bit is cleared. The application must wait until the VBUS discharges to 0.2 V, after the B-Session Valid bit in this register (USB_GOTGCTL.BSESVLD) is cleared. This discharge time can be obtained from the datasheet.			
0	SESREQSCS	0	R	**Session Request Success (device only)**
	The core sets this bit when a session request initiation is successful.			

AG.6.9 USB_GOTGINT - OTG Interrupt Register

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

Offset	Bit Position																														
0x3C004	31	30	29	28	27	26	25	24	23	22	21	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													RW1H	RW1H	RW1H								RW1H	RW1H						RW1H		
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.		
19	DBNCEDONE	0	RW1H	**Debounce Done (host only)**
	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 (USB_GUSBCFG.HNPCAP or USB_GUSBCFG.SRPCAP, respectively). This bit can be set only by the core and the application should write 1 to clear it.			
18	ADEVTOUTCHG	0	RW1H	**A-Device Timeout Change (host and device)**
	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	RW1H	**Host Negotiation Detected (host and device)**
	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.		
9	HSTNEGSUCSTSCHNG	0	RW1H	**Host Negotiation Success Status Change (host and device)**
	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 (USB_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	SESREQSUCSTSCHNG	0	RW1H	**Session Request Success Status Change (host and device)**
	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 (USB_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.		
2	SESENDDET	0	RW1H	**Session End Detected (host and device)**
	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.		

AG.6.10 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																													
0x3C008	31	30	29	28	27	26	25	24	23	22	21	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
Access										RW	RW													RW	RW		RW	RW				RW
Name										NOTIALLDMAWRIT	REMMEMSUPP													PTXFEMPLVL	NPTXFEMPLVL		DMAEN	HBSTLEN				GLBLINTRMSK

Bit	Name	Reset	Access	Description
31:23	Reserved	To ensure compatibility with future devices, always write bits to 0.		
22	NOTIALLDMAWRIT	0	RW	**Notify All DMA Writes**
	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 USB_GAHBCFG.REMMEMSUPP is set to 1. When set, the 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. When cleared, the 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. When set, 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. When cleared, 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.			
20:9	Reserved	To ensure compatibility with future devices, always write bits to 0.		
8	PTXFEMPLVL	0	RW	**Periodic TxFIFO Empty Level (host only)**
	Indicates when the Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register (USB_GINTSTS.PTXFEMP) is triggered. This bit is used only in Slave mode.			

Value	Mode	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 (host and device)**
	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.		
5	DMAEN	0	RW	**DMA Enable (host and device)**
	When set to 0 the core operates in Slave mode. When set to 1 the core operates in a DMA mode.			
4:1	HBSTLEN	0x0	RW	**Burst Length/Type (host and device)**
	This field is used in DMA mode.			

Bit	Name	Reset	Access	Description

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.

Bit	Name	Reset	Access	Description
0	GLBLINTRMSK	0	RW	**Global Interrupt Mask (host and device)**
	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. Set to unmask.			

AG.6.11 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																													
0x3C00C	31	30	29	28	27	26	25	24	23	22	21	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 (host and device)**
	Writing a 1 to this bit forces the core to device mode irrespective of the state of the ID pin. 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 (host and device)**
	Writing a 1 to this bit forces the core to host mode irrespective of the state of the ID pin. After setting the force bit, the application must wait at least 65 ms before the change to take effect.			
28	TXENDDELAY	0	RW	**Tx End Delay (device only)**
	Writing 1 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.			
27:23	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
22	TERMSELDLPULSE	0	RW	**TermSel DLine Pulsing Selection (device only)**
	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 [?]		
13:10	USBTRDTIM	0x5	RW	**USB Turnaround Time (device only)**
	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 (host and device)**
	The application uses this bit to control the core's HNP capabilities. Set to enable HNP capability.			
8	SRPCAP	0	RW	**SRP-Capable (host and device)**
	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 [?]		
5	FSINTF	0	RW	**Full-Speed Serial Interface Select (host and device)**

Bit	Name	Reset	Access	Description
	Always write this bit to 0.			
4:3	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
2:0	TOUTCAL	0x0	RW	**Timeout Calibration (host and device)**
	Always write this field to 0.			

AG.6.12 USB_GRSTCTL - Reset Register

The application uses this register to reset various hardware features inside the core.

Offset	Bit Position																												
0x3C010	31	30	29	28	27	26	25	24	23	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
Access	R	R																				RW					RW1H	RW1H		RW1H		RW1H
Name	AHBIDLE	DMAREQ																				TXFNUM					TXFFLSH	RXFFLSH		FRMCNTRRST		CSFTRST

Bit	Name	Reset	Access	Description
31	AHBIDLE	1	R	**AHB Master Idle (host and device)**
	Indicates that the AHB Master State Machine is in the IDLE condition.			
30	DMAREQ	0	R	**DMA Request Signal (host and device)**
	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 [?]		
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	RW1H	**TxFIFO Flush (host and device)**
	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.			
4	RXFFLSH	0	RW1H	**RxFIFO Flush (host and device)**
	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 [?]		
2	FRMCNTRRST	0	RW1H	**Host Frame Counter Reset (host only)**
	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	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
0	CSFTRST	0	RW1H	**Core Soft Reset (host and device)**

Bit	Name	Reset	Access	Description
				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.

AG.6.13 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 RW1H, 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																													
0x3C014	31	30	29	28	27	26	25	24	23	22	21	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	1	0	0	0	0	0
Access	RW1H	RW1H	RW1H	RW1H	R	R	R	R	RW1H	RW1H	RW1H	RW1H	HR	R				RW1H	RW1H	RW1H	RW1H	RW1H			R	R	R	R	RW1H		RW1H	
Name	WKUPINT	SESSREQINT	DISCONNINT	CONIDSTSCHNG		PTXFEMP	HCHINT	PRTINT	RESETDET	FETSUSP	INCOMPLP	INCOMPISOIN	OEPINT	IEPINT				ISOOUTDROP	ENUMDONE	USBRST	USBSUSP	ERLYSUSP			GOUTNAKEFF	GINNAKEFF	NPTXFEMP	RXFLVL	SOF	OTGINT	MODEMIS	CURMOD

Bit	Name	Reset	Access	Description
31	WKUPINT	0	RW1H	**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	RW1H	**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	RW1H	**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	RW1H	**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 [?]		
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.			
23	RESETDET	0	RW1H	**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	RW1H	**Data Fetch Suspended (device only)**

Bit	Name	Reset	Access	Description
	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	RW1H	**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	RW1H	**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.			
17:15	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
14	ISOOUTDROP	0	RW1H	**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	RW1H	**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.			
12	USBRST	0	RW1H	**USB Reset (device only)**
	The core sets this bit to indicate that a reset is detected on the USB.			
11	USBSUSP	0	RW1H	**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	RW1H	**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 [?]		
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	RW1H	**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	RW1H	**Mode Mismatch Interrupt (host and device)**

Bit	Name	Reset	Access	Description
	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.

AG.6.14 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																													
0x3C018	31	30	29	28	27	26	25	24	23	22	21	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	WKUPINTMSK	SESSREQINTMSK	DISCONNINTMSK	CONIDSTSCHNGMSK		PTXFEMPMSK	HCHINTMSK	PRTINTMSK	RESETDETMSK	FETSUSPMSK	INCOMPLPMSK	INCOMPISOINMSK	OEPINTMSK	IEPINTMSK				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 [?]		
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.			
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.			

Bit	Name	Reset	Access	Description
17:15	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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 [?]		
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.			
0	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		

AG.6.15 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																															
0x3C01C	31	30	29	28	27	26	25	24	23	22	21	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											CHEPNUM			

Bit	Name	Reset	Access	Description
31:28	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
27:24	FN	0x0	R	**Frame Number (device only)**
	This is the least significant 4 bits of the Frame number in which the packet is received on the USB.			
23:21	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
20:17	PKTSTS	0x0	R	**Packet Status (host or device)**
	Indicates the status of the received packet.			

Bit	Name	Reset	Access	Description

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 (host or device)**

Host mode: Indicates the byte count of the received IN data packet. Device mode: Indicates the byte count of the received data packet.

3:0 CHEPNUM 0x0 R **Channel Number (host only) / Endpoint Number (device only)**

Host mode: Indicates the channel number to which the current received packet belongs. Device mode: Indicates the endpoint number to which the current received packet belongs.

AG.6.16 USB_GRXSTSP - Receive Status Read and 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																															
0x3C020	31	30	29	28	27	26	25	24	23	22	21	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											CHEPNUM			

Bit	Name	Reset	Access	Description
31:25	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
24:21	FN	0x0	R	**Frame Number (device only)**

This is the least significant 4 bits of the Frame number in which the packet is received on the USB.

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.

Bit	Name	Reset	Access	Description
	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 (host or device)**
	Host mode: Indicates the byte count of the received IN data packet. Device mode: Indicates the byte count of the received data packet.			
3:0	CHEPNUM	0x0	R	**Channel Number (host only) / Endpoint Number (device only)**
	Host mode: Indicates the channel number to which the current received packet belongs. Device mode: Indicates the endpoint number to which the current received packet belongs.			

AG.6.17 USB_GRXFSIZ - Receive FIFO Size Register

The application can program the RAM size that must be allocated to the RxFIFO.

Offset	Bit Position																														
0x3C024	31	30	29	28	27	26	25	24	23	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 [?]		
9:0	RXFDEP	0x200	RW	**RxFIFO Depth**
	This value is in terms of 32-bit words. Minimum value is 16. Maximum value is 512.			

AG.6.18 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																														
0x3C028	31	30	29	28	27	26	25	24	23	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																0x200															
Access	RW																RW															
Name	NPTXFINEPTXF0DEP																NPTXFSTADDR															

Bit	Name	Reset	Access	Description
31:16	NPTXFINEPTXF0DEP	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.			
15:10	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
9:0	NPTXFSTADDR	0x200	RW	**Non-periodic Transmit RAM Start Address (host only)**
	This field contains the memory start address for Non-periodic Transmit FIFO RAM. Programmed values must not exceed the reset value.			

AG.6.19 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																															
0x3C02C	31	30	29	28	27	26	25	24	23	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 [?]		
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 Available**
	Indicates the amount of free space available in the Non-periodic TxFIFO. Values are in terms of 32-bit words.			

AG.6.20 USB_GDFIFOCFG - Global DFIFO Configuration Register

Offset	Bit Position																															
0x3C05C	31	30	29	28	27	26	25	24	23	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	**Endpoint Info Base Address**
	This field provides the start address of the EP info controller.			
15:0	GDFIFOCFG	0x0200	RW	**DFIFO Config**
	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.			

AG.6.21 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

Offset	Bit Position																													
0x3C100	31	30	29	28	27	26	25	24	23	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 [?]		
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 [?]		
10:0	PTXFSTADDR	0x400	RW	**Host Periodic TxFIFO Start Address**
	This field contains the memory start address for Host Periodic TxFIFO.			

AG.6.22 USB_DIEPTXF1 - Device IN Endpoint Transmit FIFO 1 Size Register

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																													
0x3C104	31	30	29	28	27	26	25	24	23	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 [?]		
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 [?]		
10:0	INEPNTXFSTADDR	0x400	RW	**IN Endpoint FIFO 1 Transmit RAM Start Address**
	This field contains the memory start address for IN endpoint Transmit FIFO 1.			

AG.6.23 USB_DIEPTXF2 - Device IN Endpoint Transmit FIFO 2 Size Register

This register holds the size and memory start address of IN endpoint TxFIFO 2 in Device mode. For IN endpoint FIFO 0 use USB_GNPTXFSIZ register for programming the size and memory start address.

Offset	Bit Position																													
0x3C108	31	30	29	28	27	26	25	24	23	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										

Offset	Bit Position			
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 [?]		
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 [?]		
10:0	INEPNTXFSTADDR	0x600	RW	**IN Endpoint FIFO 2 Transmit RAM Start Address**
	This field contains the memory start address for IN endpoint Transmit FIFO 2.			

AG.6.24 USB_DIEPTXF3 - Device IN Endpoint Transmit FIFO 3 Size Register

This register holds the size and memory start address of IN endpoint TxFIFO 3 in Device mode. For IN endpoint FIFO 0 use USB_GNPTXFSIZ register for programming the size and memory start address.

Offset	Bit Position																													
0x3C10C	31	30	29	28	27	26	25	24	23	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 [?]		
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 [?]		
11:0	INEPNTXFSTADDR	0x800	RW	**IN Endpoint FIFO 3 Transmit RAM Start Address**
	This field contains the memory start address for IN endpoint Transmit FIFO 3.			

AG.6.25 USB_DIEPTXF4 - Device IN Endpoint Transmit FIFO 4 Size Register

This register holds the size and memory start address of IN endpoint TxFIFO 4 in Device mode. For IN endpoint FIFO 0 use USB_GNPTXFSIZ register for programming the size and memory start address.

Offset	Bit Position																													
0x3C110	31	30	29	28	27	26	25	24	23	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											

Offset	Bit Position			
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 [?]		
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 [?]		
11:0	INEPNTXFSTADDR	0xA00	RW	**IN Endpoint FIFO 4 Transmit RAM Start Address**
	This field contains the memory start address for IN endpoint Transmit FIFO 4.			

AG.6.26 USB_DIEPTXF5 - Device IN Endpoint Transmit FIFO 5 Size Register

This register holds the size and memory start address of IN endpoint TxFIFO 5 in Device mode. For IN endpoint FIFO 0 use USB_GNPTXFSIZ register for programming the size and memory start address.

Offset	Bit Position																													
0x3C114	31	30	29	28	27	26	25	24	23	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 [?]		
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 [?]		
11:0	INEPNTXFSTADDR	0xC00	RW	**IN Endpoint FIFO 5 Transmit RAM Start Address**
	This field contains the memory start address for IN endpoint Transmit FIFO 5.			

AG.6.27 USB_DIEPTXF6 - Device IN Endpoint Transmit FIFO 6 Size Register

This register holds the size and memory start address of IN endpoint TxFIFO 6 in Device mode. For IN endpoint FIFO 0 use USB_GNPTXFSIZ register for programming the size and memory start address.

Offset	Bit Position																													
0x3C118	31	30	29	28	27	26	25	24	23	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											

Offset	Bit Position			
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 [?]		
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 [?]		
11:0	INEPNTXFSTADDR	0xE00	RW	**IN Endpoint FIFO 6 Transmit RAM Start Address**
	This field contains the memory start address for IN endpoint Transmit FIFO 6.			

AG.6.28 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																													
0x3C400	31	30	29	28	27	26	25	24	23	22	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					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 [?]		
15:8	RESVALID	0x00	RW	**Resume Validation Period**
	This field is effective only when USB_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 USB_HCFG.ENA32KHZS is set.			
7	ENA32KHZS	0	RW	**Enable 32 KHz Suspend mode**
	When this bit is set the core expects that the clock to the core 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 [?]		
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.

Bit	Name	Reset	Access	Description
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).

AG.6.29 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																															
0x3C404	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset																0	0x17D7															
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 [?]		
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.	
15:0	FRINT	0x17D7	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.			

AG.6.30 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																															
0x3C408	31	30	29	28	27	26	25	24	23	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, in terms of PHY clocks. This field decrements on each PHY 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 0x3FFF.			

AG.6.31 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

Offset	Bit Position																															
0x3C410	31	30	29	28	27	26	25	24	23	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.			

AG.6.32 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																															
0x3C414	31	30	29	28	27	26	25	24	23	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 [?]		
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.			

AG.6.33 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																															
0x3C418	31	30	29	28	27	26	25	24	23	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													

Offset	Bit Position	
Name		HAINTMSK

Bit	Name	Reset	Access	Description
31:14	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
13:0	HAINTMSK	0x0000	RW	**Channel Interrupt Mask for channel 0 - 13**
	Set bit n to unmask channel n interrupts.			

AG.6.34 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 RW1H bits, the application must write a 1 to the bit to clear the interrupt.

Offset	Bit Position																												
0x3C440	31	30	29	28	27	26	25	24	23	22	21	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	RW1H	RW	RW1H	R	RW1H	RW1H	RW1H	RW1H
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 [?]		
18:17	PRTSPD	0x0	R	**Port Speed**
	Indicates the speed of the device attached to this port.			

Value	Mode	Description
0	HS	High speed.
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 [0]: Logic level of D+. Bit [1]: Logic level of D-.			
9	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		

Bit	Name	Reset	Access	Description
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	RW1H	**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	PRTOVRCURRCHNG	0	RW1H	**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. When there is an overcurrent condition this bit is 1.			
3	PRTENCHNG	0	RW1H	**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	RW1H	**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	PRTCONNDET	0	RW1H	**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 (USB_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**
	When this bit is 1 a device is attached to the port.			

AG.6.35 USB_HCx_CHAR - Host Channel x Characteristics Register

Offset	Bit Position																														
0x3C500	31	30	29	28	27	26	25	24	23	22	21	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	RW1H	RW1H	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	RW1H	**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	RW1H	**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.			
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**

Bit	Name	Reset	Access	Description
	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 [?]		
15	EPDIR	0	RW	**Endpoint Direction**
	Indicates whether the transaction is IN or OUT.			

Value	Mode	Description
0	OUT	Direction is OUT.
1	IN	Direction is IN.

Bit	Name	Reset	Access	Description
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.			

AG.6.36 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																											
0x3C508	31	30	29	28	27	26	25	24	23	22	21	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																						RW1H	RW1H	RW1H	RW1H		RW1H	RW1H	RW1H	RW1H	RW1H	RW1H
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 [?]		
10	DATATGLERR	0	RW1H	**Data Toggle Error**
	This bit can be set only by the core and the application should write 1 to clear it.			
9	FRMOVRUN	0	RW1H	**Frame Overrun**
	This bit can be set only by the core and the application should write 1 to clear it.			
8	BBLERR	0	RW1H	**Babble Error**
	This bit can be set only by the core and the application should write 1 to clear it.			
7	XACTERR	0	RW1H	**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 [?]		
5	ACK	0	RW1H	**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	RW1H	**NAK Response Received Interrupt**

Bit	Name	Reset	Access	Description
	This bit can be set only by the core and the application should write 1 to clear it.			
3	STALL	0	RW1H	**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	RW1H	**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	RW1H	**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	RW1H	**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.			

AG.6.37 USB_HCx_INTMSK - Host Channel x Interrupt Mask Register

This register reflects the mask for each channel status described in the USB_CHx_INT.

Offset	Bit Position																											
0x3C50C	31	30	29	28	27	26	25	24	23	22	21	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 [?]		
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 [?]		
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.			

AG.6.38 USB_HCx_TSIZ - Host Channel x Transfer Size Register

Offset	Bit Position																															
0x3C510	31	30	29	28	27	26	25	24	23	22	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 [?]		
30:29	PID	0x0	RW	**Packet ID**
	The application programs this field with the packet ID type to use for 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.

Bit	Name	Reset	Access	Description
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.			
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).			

AG.6.39 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. The starting DMA address must be DWORD-aligned.

Offset	Bit Position																															
0x3C514	31	30	29	28	27	26	25	24	23	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	DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DMAADDR	0xXXXXXXXX	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).			

AG.6.40 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																															
0x3C800	31	30	29	28	27	26	25	24	23	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																			0x0		0x00							0	0	0x0	

Offset	Bit Position							
Access	RW		RW	RW		RW	RW	RW
Name	RESVALID		PERFRINT	DEVADDR		ENA32KHZSUSP	NZSTSOUTHSHK	DEVSPD

Bit	Name	Reset	Access	Description
31:26	RESVALID	0x02	RW	**Resume Validation Period**
	This field is effective only when USB_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 USB_DCFG.ENA32KHZSUSP is set.			
25:13	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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.			
3	ENA32KHZSUSP	0	RW	**Enable 32 KHz Suspend mode**
	When this bit is set, the core expects that the PHY 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. When set to 1 send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application. When set to 0 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).

AG.6.41 USB_DCTL - Device Control Register

Offset	Bit Position																												
0x3C804	31	30	29	28	27	26	25	24	23	22	21	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	0	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 [?]		
16	NAKONBBLE	0	RW	**NAK on Babble Error**

Bit	Name	Reset	Access	Description
	Set NAK automatically on babble. The core sets NAK automatically for the endpoint on which babble is received.			
15	IGNRFRMNUM	0	RW	**Ignore Frame number For Isochronous End points**
	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 [?]		
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 this bit only after making sure that the Global OUT NAK Effective bit in the Core Interrupt Register (USB_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 application must set this bit only after making sure that the Global IN NAK Effective bit in the Core Interrupt Register (USB_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.
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**
	When this bit is 0 a handshake is sent based on the FIFO Status and the NAK and STALL bit settings. When this bit is 1 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**
	When this bit is 0 a handshake is sent out based on the data availability in the transmit FIFO. When this bit is 1 a NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.			
1	SFTDISCON	0	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.			

AG.6.42 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																														
0x3C808	31	30	29	28	27	26	25	24	23	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																		0	0x1		0
Access											R																		R	R		R
Name											SOFFN																		ERRTICERR	ENUMSPD		SUSPSTS

Bit	Name	Reset	Access	Description
31:22	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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 [?]		
3	ERRTICERR	0	R	**Erratic Error**
	The core sets this bit to report any erratic errors (PHY error). Due to erratic errors, the core goes into Suspended state and an interrupt is generated to the application with. Early Suspend bit of the Core Interrupt register (USB_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 bus for an extended period of time. The core comes out of the suspend when there is any activity on the bus or when the application writes to the Remote Wakeup Signaling bit in the Device Control register (USB_DCTL.RMTWKUPSIG).			

AG.6.43 USB_DIEPMSK - Device IN Endpoint Common Interrupt Mask Register

This register works with each of the Device IN Endpoint Interrupt (USB_DIEP0INT/USB_DIEPx_INT) registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the USB_DIEP0INT/USB_DIEPx_INT register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

Offset	Bit Position																														
0x3C810	31	30	29	28	27	26	25	24	23	22	21	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																			NAKMSK					TXFIFOUNDRNMSK		INEPNAKEFFMSK		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 [?]		
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 [?]		
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 [?]		
6	INEPNAKEFFMSK	0	RW	**IN Endpoint NAK Effective Mask**
	Set to 1 to unmask INEPNAKEFF Interrupt.			
5	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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.			

Bit	Name	Reset	Access	Description
0	XFERCOMPLMSK	0	RW	**Transfer Completed Interrupt Mask**
	Set to 1 to unmask XFERCOMPL Interrupt.			

AG.6.44 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																														
0x3C814	31	30	29	28	27	26	25	24	23	22	21	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	BBLEERRMSK				OUTPKTERRMSK		BACK2BACKSETUP		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 [?]		
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 [?]		
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 [?]		
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	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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.			
0	XFERCOMPLMSK	0	RW	**Transfer Completed Interrupt Mask**
	Set to 1 to unmask XFERCOMPL Interrupt.			

AG.6.45 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																														
0x3C818	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Offset	Bit Position																													
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 [?]		
22	OUTEPINT6	0	R	**OUT Endpoint 6 Interrupt Bit**
	This bit is set when on or more of the interrupt flags in USB_DOEP5_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_DOEP4_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_DOEP3_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_DOEP2_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_DOEP1_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_DOEP0_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 [?]		
6	INEPINT6	0	R	**IN Endpoint 6 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP5_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_DIEP4_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_DIEP3_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_DIEP2_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_DIEP1_INT are set.			
1	INEPINT1	0	R	**IN Endpoint 1 Interrupt Bit**
	This bit is set when one or more of the interrupt flags in USB_DIEP0_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.			

AG.6.46 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																													
0x3C81C	31	30	29	28	27	26	25	24	23	22	21	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 [?]		
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 [?]		
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.			
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.			

AG.6.47 USB_DVBUSDIS - Device VBUS Discharge Time Register

This register specifies the VBUS discharge time after VBUS pulsing during SRP.

Offset	Bit Position																														
0x3C828	31	30	29	28	27	26	25	24	23	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 [?]		
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.			

AG.6.48 USB_DVBUSPULSE - Device VBUS Pulsing Time Register

This register specifies the VBUS pulsing time during SRP.

Offset	Bit Position																															
0x3C82C	31	30	29	28	27	26	25	24	23	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 [?]		
11:0	DVBUSPULSE	0x5B8	RW	**Device VBUS Pulsing Time**
	Specifies the VBUS pulsing time during SRP. This value equals VBUS pulsing time in PHY clocks / 1024.			

AG.6.49 USB_DIEPEMPMSK - Device IN Endpoint FIFO Empty Interrupt Mask Register

This register is used to control the IN endpoint FIFO empty interrupt generation (USB_DIEP0INT/USB_DIEPx_INT.TXFEMP).

Offset	Bit Position																															
0x3C834	31	30	29	28	27	26	25	24	23	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	DIEPEMPMSK																															

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
15:0	DIEPEMPMSK	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.			

AG.6.50 USB_DIEP0CTL - Device IN Endpoint 0 Control Register

This section describes the Control IN Endpoint 0 Control register. Nonzero control endpoints use registers for endpoints 1 - 6.

Offset	Bit Position																															
0x3C900	31	30	29	28	27	26	25	24	23	22	21	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	RW1H	RW1H			W1	W1	RW				RW1H		R		R		R														RW	
Name	EPENA	EPDIS			SNAK	CNAK	TXFNUM				STALL		EPTYPE		NAKSTS		USBACTEP														MPS	

Bit	Name	Reset	Access	Description
31	EPENA	0	RW1H	**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	RW1H	**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 [?]		
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**
	This value is set to the FIFO number that is assigned to IN Endpoint 0.			
21	STALL	0	RW1H	**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 [?]		
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 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 [?]		
15	USBACTEP	1	R	**USB Active Endpoint**
	This bit is always 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 [?]		
1:0	MPS	0x0	RW	**Maximum Packet Size**
	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.

AG.6.51 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																													
0x3C908	31	30	29	28	27	26	25	24	23	22	21	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
Access																			RW1H	RW1H	RW1H				R	RW1H		RW1H	RW1H	RW1H	RW1H	RW1H
Name																			NAKINTRPT	BBLEERR	PKTDRPSTS				TXFEMP	INEPNAKEFF		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 [?]		

Bit	Name	Reset	Access	Description
13	NAKINTRPT	0	RW1H	**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	RW1H	**NAK Interrupt**
	The core generates this interrupt when babble is received for the endpoint.			
11	PKTDRPSTS	0	RW1H	**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.			
10:8	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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	RW1H	**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	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
4	INTKNTXFEMP	0	RW1H	**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.			
3	TIMEOUT	0	RW1H	**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	RW1H	**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	RW1H	**Endpoint Disabled Interrupt**
	This bit indicates that the endpoint is disabled per the application's request.			
0	XFERCOMPL	0	RW1H	**Transfer Completed Interrupt**
	This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.			

AG.6.52 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																														
0x3C910	31	30	29	28	27	26	25	24	23	22	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 [?]		
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 [?]		
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.			

AG.6.53 USB_DIEP0DMAADDR - Device IN Endpoint 0 DMA Address Register

Offset	Bit Position																															
0x3C914	31	30	29	28	27	26	25	24	23	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	DIEP0DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DIEP0DMAADDR	0xXXXXXXXX	RW	**DMA Address**
	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).			

AG.6.54 USB_DIEP0TXFSTS - Device IN Endpoint 0 Transmit FIFO Status Register

This read-only register contains the free space information for the Device IN endpoint 0 TxFIFO.

Offset	Bit Position																															
0x3C918	31	30	29	28	27	26	25	24	23	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 [?]		
15:0	SPCAVAIL	0x0200	R	**TxFIFO Space Available**
	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words.			

AG.6.55 USB_DIEPx_CTL - Device 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																															
0x3C920	31	30	29	28	27	26	25	24	23	22	21	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	RW1H	RW1H	W1	W1	W1	W1	RW				RW1H		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	RW1H	**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	RW1H	**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	RW1H	**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.			
20	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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.

Bit	Name	Reset	Access	Description
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.

Bit	Name	Reset	Access	Description
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 [?]		
10:0	MPS	0x000	RW	**Maximum Packet Size**

Bit	Name	Reset	Access	Description
	The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.			

AG.6.56 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																													
0x3C928	31	30	29	28	27	26	25	24	23	22	21	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
Access																			RW1H	RW1H	RW1H				R	RW1H		RW1H	RW1H	RW1H	RW1H	RW1H
Name																			NAKINTRPT	BBLEERR	PKTDRPSTS				TXFEMP	INEPNAKEFF		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 [?]		
13	NAKINTRPT	0	RW1H	**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	RW1H	**NAK Interrupt**
	The core generates this interrupt when babble is received for the endpoint.			
11	PKTDRPSTS	0	RW1H	**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.			
10:8	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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	RW1H	**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	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
4	INTKNTXFEMP	0	RW1H	**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.			
3	TIMEOUT	0	RW1H	**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	RW1H	**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	RW1H	**Endpoint Disabled Interrupt**
	This bit indicates that the endpoint is disabled per the application's request.			
0	XFERCOMPL	0	RW1H	**Transfer Completed Interrupt**
	This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.			

AG.6.57 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																															
0x3C930	31	30	29	28	27	26	25	24	23	22	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 [?]		
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.			

AG.6.58 USB_DIEPx_DMAADDR - Device IN Endpoint x+1 DMA Address Register

Offset	Bit Position																															
0x3C934	31	30	29	28	27	26	25	24	23	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	DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DMAADDR	0xXXXXXXXX	RW	**DMA Address**
	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).			

AG.6.59 USB_DIEPx_TXFSTS - Device IN Endpoint x+1 Transmit FIFO Status Register

This read-only register contains the free space information for the Device IN endpoint TxFIFO.

Offset	Bit Position																															
0x3C938	31	30	29	28	27	26	25	24	23	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															

Offset	Bit Position
Name	SPCAVAIL

Bit	Name	Reset	Access	Description
31:16	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
15:0	SPCAVAIL	0x0200	R	**TxFIFO Space Available**
	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words.			

AG.6.60 USB_DOEP0CTL - Device 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																													
0x3CB00	31	30	29	28	27	26	25	24	23	22	21	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	RW1H	R			W1	W1					RW1H	RW	R		R		R														R	
Name	EPENA	EPDIS			SNAK	CNAK					STALL	SNP	EPTYPE		NAKSTS		USBACTEP														MPS	

Bit	Name	Reset	Access	Description
31	EPENA	0	RW1H	**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 [?]		
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 [?]		
21	STALL	0	RW1H	**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 [?]		
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 [?]		
1:0	MPS	0x0	R	**Maximum Packet Size**

Bit	Name	Reset	Access	Description
	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.

AG.6.61 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																												
0x3CB08	31	30	29	28	27	26	25	24	23	22	21	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																			RW1H	RW1H	RW1H					RW1H		RW1H	RW1H	RW1H	RW1H	RW1H
Name																			NAKINTRPT	BBLEERR	PKTDRPSTS					BACK2BACKSETUP		OUTTKNEPDIS	SETUP	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 [?]		
13	NAKINTRPT	0	RW1H	**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	RW1H	**NAK Interrupt**
	The core generates this interrupt when babble is received for the endpoint.			
11	PKTDRPSTS	0	RW1H	**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.			
10:7	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
6	BACK2BACKSETUP	0	RW1H	**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.			
5	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
4	OUTTKNEPDIS	0	RW1H	**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	RW1H	**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	RW1H	**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	RW1H	**Endpoint Disabled Interrupt**
	This bit indicates that the endpoint is disabled per the application's request.			
0	XFERCOMPL	0	RW1H	**Transfer Completed Interrupt**
	This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.			

AG.6.62 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																														
0x3CB10	31	30	29	28	27	26	25	24	23	22	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 [?]		
30:29	SUPCNT	0x0	RW	**SETUP Packet Count**
	This field specifies the number of back-to-back SETUP data packets the endpoint can receive.			
28:20	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
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 [?]		
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.			

AG.6.63 USB_DOEP0DMAADDR - Device OUT Endpoint 0 DMA Address Register

Offset	Bit Position																														
0x3CB14	31	30	29	28	27	26	25	24	23	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	DOEP0DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DOEP0DMAADDR	0xXXXXXXXX	RW	**DMA Address**
	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).			

AG.6.64 USB_DOEPx_CTL - Device 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																														
0x3CB20	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Offset	Bit Position																
Reset	0	0	0	0	0	0		0	0	0x0	0	0	0		0x000		
Access	RW1H	RW1H	W1	W1	W1	W1		RW1H	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	RW1H	**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	RW1H	**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 [?]		
21	STALL	0	RW1H	**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.

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.

Bit	Name	Reset	Access	Description
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 [?]		
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.			

AG.6.65 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																															
0x3CB28	31	30	29	28	27	26	25	24	23	22	21	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																			RW1H	RW1H	RW1H					RW1H		RW1H	RW1H	RW1H	RW1H	RW1H
Name																			NAKINTRPT	BBLEERR	PKTDRPSTS					BACK2BACKSETUP		OUTTKNEPDIS	SETUP	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 [?]		
13	NAKINTRPT	0	RW1H	**NAK Interrupt**
	The core generates this interrupt when a NAK is transmitted or received by the device.			
12	BBLEERR	0	RW1H	**Babble Error**
	The core generates this interrupt when babble is received for the endpoint.			
11	PKTDRPSTS	0	RW1H	**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.			
10:7	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
6	BACK2BACKSETUP	0	RW1H	**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.			
5	Reserved	To ensure compatibility with future devices, always write bits to 0. More information in [?]		
4	OUTTKNEPDIS	0	RW1H	**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	RW1H	**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	RW1H	**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	RW1H	**Endpoint Disabled Interrupt**
	This bit indicates that the endpoint is disabled per the application's request.			
0	XFERCOMPL	0	RW1H	**Transfer Completed Interrupt**
	This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.			

AG.6.66 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																															
0x3CB30	31	30	29	28	27	26	25	24	23	22	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 [?]		
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.

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.			

AG.6.67 USB_DOEPx_DMAADDR - Device OUT Endpoint x+1 DMA Address Register

Offset	Bit Position																															
0x3CB34	31	30	29	28	27	26	25	24	23	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	DMAADDR																															

Bit	Name	Reset	Access	Description
31:0	DMAADDR	0xXXXXXXXX	RW	**DMA Address**
	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).			

AG.6.68 USB_PCGCCTL - Power and Clock Gating Control Register

This register is available in Host and Device modes. The application use this register to control the core's power-down and clock gating features.

Offset	Bit Position																															
0x3CE00	31	30	29	28	27	26	25	24	23	22	21	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 [?]		
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 [?]		
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 [?]		
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.			

AG.6.69 USB_FIFO0Dx - Device EP 0/Host Channel 0 FIFO

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.

Offset	Bit Position																															
0x3D000	31	30	29	28	27	26	25	24	23	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	FIFO0D																															

Bit	Name	Reset	Access	Description
31:0	FIFO0D	0xXXXXXXXX	RW	**Device EP 0/Host Channel 0 FIFO**
	FIFO 0 push/pop region. Used in slave mode.			

AG.6.70 USB_FIFO1Dx - Device EP 1/Host Channel 1 FIFO

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.

Offset	Bit Position																														
0x3E000	31	30	29	28	27	26	25	24	23	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	FIFO1D																															

Bit	Name	Reset	Access	Description
31:0	FIFO1D	0xXXXXXXXX	RW	**Device EP 1/Host Channel 1 FIFO**
	FIFO 1 push/pop region. Used in slave mode.			

AG.6.71 USB_FIFO2Dx - Device EP 2/Host Channel 2 FIFO

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.

Offset	Bit Position																														
0x3F000	31	30	29	28	27	26	25	24	23	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	FIFO2D																															

Bit	Name	Reset	Access	Description
31:0	FIFO2D	0xXXXXXXXX	RW	**Device EP 2/Host Channel 2 FIFO**
	FIFO 2 push/pop region. Used in slave mode.			

AG.6.72 USB_FIFO3Dx - Device EP 3/Host Channel 3 FIFO

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.

Offset	Bit Position																														
0x40000	31	30	29	28	27	26	25	24	23	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	FIFO3D																															

Bit	Name	Reset	Access	Description
31:0	FIFO3D	0xXXXXXXXX	RW	**Device EP 3/Host Channel 3 FIFO**
	FIFO 3 push/pop region. Used in slave mode.			

AG.6.73 USB_FIFO4Dx - Device EP 4/Host Channel 4 FIFO

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.

Offset	Bit Position																														
0x41000	31	30	29	28	27	26	25	24	23	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	FIFO4D																															

Bit	Name	Reset	Access	Description
31:0	FIFO4D	0xXXXXXXXX	RW	**Device EP 4/Host Channel 4 FIFO**
	FIFO 4 push/pop region. Used in slave mode.			

AG.6.74 USB_FIFO5Dx - Device EP 5/Host Channel 5 FIFO

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.

Offset	Bit Position																														
0x42000	31	30	29	28	27	26	25	24	23	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	FIFO5D																															

Bit	Name	Reset	Access	Description
31:0	FIFO5D	0xXXXXXXXX	RW	**Device EP 5/Host Channel 5 FIFO**
	FIFO 5 push/pop region. Used in slave mode.			

AG.6.75 USB_FIFO6Dx - Device EP 6/Host Channel 6 FIFO

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.

Offset	Bit Position																														
0x43000	31	30	29	28	27	26	25	24	23	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	FIFO6D																															

Bit	Name	Reset	Access	Description
31:0	FIFO6D	0xXXXXXXXX	RW	**Device EP 6/Host Channel 6 FIFO**
	FIFO 6 push/pop region. Used in slave mode.			

AG.6.76 USB_FIFO7Dx - Host Channel 7 FIFO

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.

Offset	Bit Position																														
0x44000	31	30	29	28	27	26	25	24	23	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	FIFO7D																															

Bit	Name	Reset	Access	Description
31:0	FIFO7D	0xXXXXXXXX	RW	**Host Channel 7 FIFO**
	FIFO 7 push/pop region. Used in slave mode.			

AG.6.77 USB_FIFO8Dx - Host Channel 8 FIFO

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.

Offset	Bit Position																														
0x45000	31	30	29	28	27	26	25	24	23	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	FIFO8D																															

Bit	Name	Reset	Access	Description
31:0	FIFO8D	0xXXXXXXXX	RW	**Host Channel 8 FIFO**
	FIFO 8 push/pop region. Used in slave mode.			

AG.6.78 USB_FIFO9Dx - Host Channel 9 FIFO

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.

Offset	Bit Position																														
0x46000	31	30	29	28	27	26	25	24	23	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	FIFO9D																															

Bit	Name	Reset	Access	Description
31:0	FIFO9D	0xXXXXXXXX	RW	**Host Channel 9 FIFO**
	FIFO 9 push/pop region. Used in slave mode.			

AG.6.79 USB_FIFO10Dx - Host Channel 10 FIFO

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.

Offset	Bit Position																														
0x47000	31	30	29	28	27	26	25	24	23	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	FIFO10D																															

Bit	Name	Reset	Access	Description
31:0	FIFO10D	0xXXXXXXXX	RW	**Host Channel 10 FIFO**
	FIFO 10 push/pop region. Used in slave mode.			

AG.6.80 USB_FIFO11Dx - Host Channel 11 FIFO

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.

Offset	Bit Position																														
0x48000	31	30	29	28	27	26	25	24	23	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	FIFO11D																															

Bit	Name	Reset	Access	Description
31:0	FIFO11D	0xXXXXXXXX	RW	**Host Channel 11 FIFO**
	FIFO 11 push/pop region. Used in slave mode.			

AG.6.81 USB_FIFO12Dx - Host Channel 12 FIFO

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.

Offset	Bit Position																														
0x49000	31	30	29	28	27	26	25	24	23	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	FIFO12D																															

Bit	Name	Reset	Access	Description
31:0	FIFO12D	0xXXXXXXXX	RW	**Host Channel 12 FIFO**
	FIFO 12 push/pop region. Used in slave mode.			

AG.6.82 USB_FIFO13Dx - Host Channel 13 FIFO

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.

Offset	Bit Position																														
0x4A000	31	30	29	28	27	26	25	24	23	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	FIFO13D																															

Bit	Name	Reset	Access	Description
31:0	FIFO13D	0xXXXXXXXX	RW	**Host Channel 13 FIFO**
	FIFO 13 push/pop region. Used in slave mode.			

AG.6.83 USB_FIFORAMx - Direct Access to Data FIFO RAM for Debugging (2 KB)

Offset	Bit Position																														
0x5C000	31	30	29	28	27	26	25	24	23	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	FIFORAM																															

Bit	Name	Reset	Access	Description
31:0	FIFORAM	0xXXXXXXXX	RW	**FIFO RAM**
	Direct Access to Data FIFO RAM for Debugging (2 KB)			

AH Device Errata

This section describes minor operational differences from the data sheet and recommended workarounds. As device and documentation issues become known, this section will be updated the document revised.

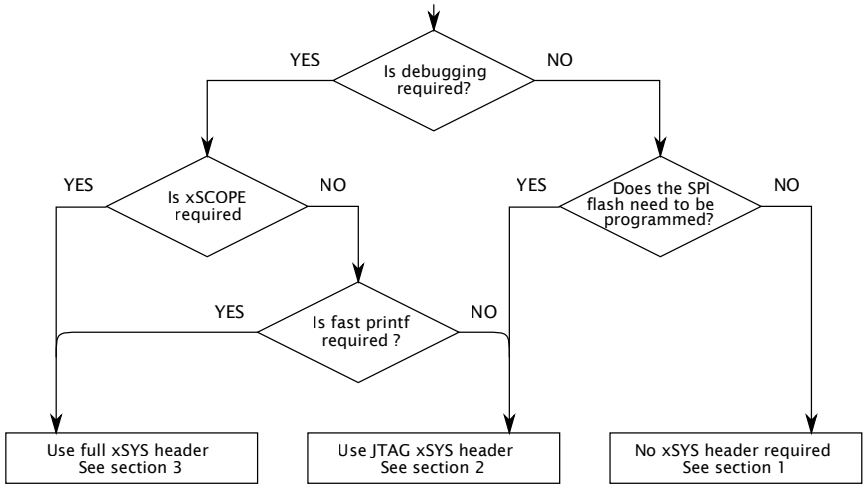
To guarantee a logic low is seen on the pins DEBUG_N, MODE[3:0], TMS, TCK and TDI, the driving circuit should present an impedance of less than 100 Ω to ground. Usually this is not a problem for CMOS drivers driving single inputs. If one or more of these inputs are placed in parallel, however, additional logic buffers may be required to guarantee correct operation.

For static inputs tied high or low, the relevant input pin should be tied directly to GND or VDDIO.

AI JTAG, xSCOPE and Debugging

If you intend to design a board that can be used with the XMOS toolchain and xTAG debugger, you will need an xSYS header on your board. Figure 356 shows a decision diagram which explains what type of xSYS connectivity you need. The three subsections below explain the options in detail.

Figure 356: Decision diagram for the xSYS header



AI.1 No xSYS header

The use of an xSYS header is optional, and may not be required for volume production designs. However, the XMOS toolchain expects the xSYS header; if you do not have an xSYS header then you must provide your own method for writing to flash/OTP and for debugging.

AI.2 JTAG-only xSYS header

The xSYS header connects to an xTAG debugger, which has a 20-pin 0.1" female IDC header. The design will hence need a male IDC header. We advise to use a boxed header to guard against incorrect plug-ins. If you use a 90 degree angled header, make sure that pins 2, 4, 6, ..., 20 are along the edge of the PCB.

Connect pins 4, 8, 12, 16, 20 of the xSYS header to ground, and then connect:

- TDI to pin 5 of the xSYS header
- TMS to pin 7 of the xSYS header
- TCK to pin 9 of the xSYS header
- DEBUG_N to pin 11 of the xSYS header
- TDO to pin 13 of the xSYS header
- RST_N to pin 15 of the xSYS header
- If MODE2 is configured high, connect MODE2 to pin 3 of the xSYS header. Do not connect to VDDIO.
- If MODE3 is configured high, connect MODE3 to pin 3 of the xSYS header. Do not connect to VDDIO.

The RST_N net should be open-drain, active-low, and have a pull-up to VDDIO.

AI.3 Full xSYS header

For a full xSYS header you will need to connect the pins as discussed in Section AI.2, and then connect a 2-wire xCONNECT Link to the xSYS header. The links can be found in the Signal description table (Section 4): they are labelled XLA, XLB, etc in the function column. The 2-wire link comprises two inputs and outputs, labelled ${}^{1}_{out}$, ${}^{0}_{out}$, ${}^{0}_{in}$, and ${}^{1}_{in}$, . For example, if you choose to use XLB of tile 0 for xSCOPE I/O, you need to connect up XLB^{1}_{out}, XLB^{0}_{out}, XLB^{0}_{in}, XLB^{1}_{in} as follows:

- XLB^{1}_{out} (X0D16) to pin 6 of the xSYS header with a 33R series resistor close to the device.
- XLB^{0}_{out} (X0D17) to pin 10 of the xSYS header with a 33R series resistor close to the device.
- XLB^{0}_{in} (X0D18) to pin 14 of the xSYS header.
- XLB^{1}_{in} (X0D19) to pin 18 of the xSYS header.

AJ Schematics Design Check List

☑ This section is a checklist for use by schematics designers using the XS1-XAU8A-10-FB265. Each of the following sections contains items to check for each design.

AJ.1 Clock

☐ Pins MODE0 and MODE1 are set to the correct value for the chosen frequency. The MODE settings are shown in the Oscillator section, Section 7. If you have a choice between two values, choose the value with the highest multiplier ratio since that will boot faster.

☐ OSC_EXT_N is tied to ground (for use with a crystal or oscillator) or tied to VDDIO (for use with the internal oscillator). If using the internal oscillator, set MODE0 and MODE1 to be for the 20-48 MHz range (Section 7).

☐ If you have used an oscillator, it is a 1V8 oscillator.

AJ.2 Boot

☐ The device is connected to a SPI flash for booting, connected to X0D0, X0D01, X0D10, and X0D11 (Section 8). If not, you must boot the xCORE Tile from the ARM-core.

☐ The device that is connected to flash has both MODE2 and MODE3 NC (Section 8).

☐ The SPI flash that you have chosen is supported by **xflash**, or you have created a specification file for it.

AJ.3 JTAG, xSCOPE, and debugging

☐ You have decided as to whether you need an xSYS header or not (Section AI)

☐ If you included an xSYS header, you connected pin 3 to any MODE2/MODE3 pin that would otherwise be NC (Section AI).

☐ If you have not included an xSYS header, you have devised a method to program the SPI-flash or OTP (Section AI).

AK PCB Layout Design Check List

☑ This section is a checklist for use by PCB designers using the XS1-XAU8A-10-FB265. Each of the following sections contains items to check for each design.

AK.1 Ground Plane

☐ Each ground ball has a via to minimize impedance and conduct heat away from the device. (Section 11.2)

☐ Other than ground vias, there are no (or only a few) vias underneath or closely around the device. This create a good, solid, ground plane.

AK.2 Power supply decoupling

☐ The decoupling capacitors are all placed close to a supply pin (Section 11).

☐ The decoupling capacitors are spaced around the device (Section 11).

☐ The ground side of each decoupling capacitor has a direct path back to the center ground of the device.

AK.3 PLLVDD

☐ The PLLVDD filter (especially the capacitor) is placed close to the PLLVDD pin (Section 11).

AL Associated Design Documentation

Document Title	Information	Document Number
XMOS Programming Guide	Timers, ports, clocks, cores and channels	X04440
xTIMEcomposer User Guide	Compilers, assembler and linker/mapper Timing analyzer, xSCOPE, debugger Flash and OTP programming utilities	X3766
xCORE-XA - Application Development		AN00141
xCORE-XA - xCORE ARM Bridge Library	xCORE-XA development	AN00142
xCORE-XA - xCORE ARM Bridge Library with DMA		AN00143
xCORE-XA - xCORE ARM Boot Library	xCORE-XA boot	AN00144
xCORE-XA - Power Management	xCORE-XA power management	AN00145

AM Related Documentation

Document Title	Information	Document Number
The XMOS XS1 Architecture	ISA manual	X7879
XS1 Port I/O Timing	Port timings	X5821
xCONNECT Architecture	Link, switch and system information	X4249
XS1-L Link Performance and Design Guidelines	Link timings	X2999
XS1-L Clock Frequency Control	Advanced clock control	X1433

AN Revision History

Date	Description
2015-01-30	Preliminary release
2015-04-20	Ball G2 set to CLK - Section 3 and Section 4 Power supply pins updated - - Section 11
2015-01-28	Added PCB Layout Design Check List - Section AK